HEVs
haute école valaisanne
hochschule wallis

HEVs2
haute école valaisanne
hochschule wallis

Filière informatique de gestion

Diplôme 2005 / 2006

Etudiant : Bruno Fernandes

Titre du diplôme :

.Net Framework 3.0 :

Windows Communication Foundation

www.hevs.ch

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale
Fachhochschule Westschweiz

Tables des matières

1. Présentation du travail de diplôme

1.1. Introduction

Le travail de diplôme a été réalisé sur une des briques du Microsoft .Net Framework 3.0. Cette brique s'appelle Windows Communication Foundation (WCF).

Les trois « Briques » de la plateforme WinFX renommée .Net Framework 3.0

1.2. Qu'est-ce que WCF ?

WCF (Windows Communication Foundation) est un nouveau type d'infrastructure de communication conçu autour de l'architecture des services Web. Une prise en charge étendue des services web garantit une messagerie basée sur les transactions sécurisées et fiables, de même qu'une totale interopérabilité.

Basé sur le .NET Framework 2.0, le modèle de programmation orienté services de WCF simplifie la conception de systèmes connectés. WCF unifie également une vaste gamme de fonctionnalités de systèmes distribués au sein d'une architecture évolutive englobant à la fois des transports, des systèmes de sécurité, des modèles de message, des codages, des topologies de réseau et des modèles d'hébergement.

Microsoft a effectué un travail considérable pour intégrer WCF aux technologies Microsoft existantes en matière de conception de systèmes distribués, notamment COM+, MSMQ et les services Web ASP.NET.

Les applications basées sur les technologies existantes peuvent désormais être exposées en tant que services sans qu'il soit nécessaire de faire des modifications considérables à ces briques logicielles. Windows Communication Foundation, utilisé au niveau de l'infrastructure, va aider de manière significative les développeurs à exposer des applications existantes en tant que service.

En outre, Windows Communication Foundation fournit des processus automatiques simples permettant de migrer des applications utilisant .NET Remoting, les services Web ASP.NET ou les .NET Enterprise Services de manière à ce qu'elles utilisent le modèle de programmation Windows Communication Foundation en mode natif.

Voici une vue globale du nouveau Framework .Net 3.0.

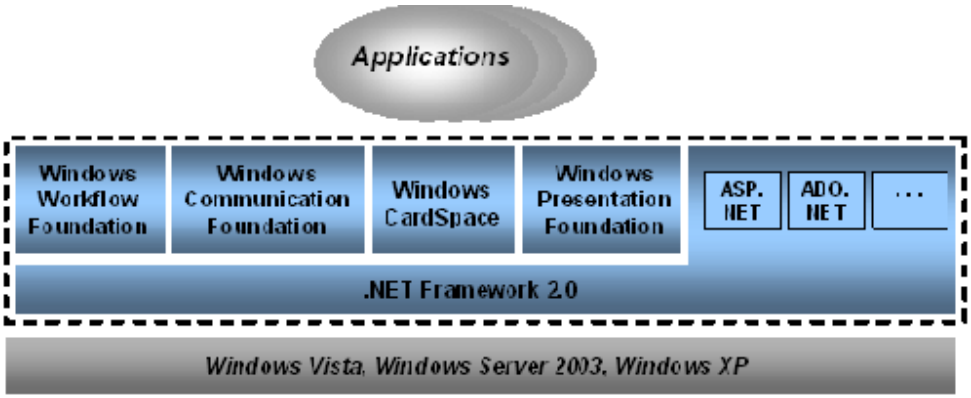


Schéma du .Net Framework 3.0

Nous pouvons voir clairement que WCF, WPF, WF & Windows Cardspace viennent s'ajouter au Framework 2.0.

1.3. Description du travail de diplôme

Le travail consiste à étudier dans les détails Windows Communication Foundation.

Après une première approche théorique (prise en main des concepts et des technologies sous-jacentes) donnant résultat à un tutoriel d'apprentissage, un développement concret d'intégration sera réalisé.

En fonction des contacts industriels pris principalement chez Nestlé, la thématique du développement sera prise durant la dernière semaine de la phase d'apprentissage.

1.4. Objectifs du travail de diplôme

Selon le descriptif de du travail de diplôme, trois principaux objectifs ont été fixés :

- Analyse des opportunités offertes par WCF en termes de développement et d'intégration
- Tutoriel d'apprentissage des principaux concepts sous-jacents
- Réalisation d'un développement concret avec WCF

1.5. Commentaire personnel

Avant le commencement de mon travail de diplôme, Windows Communication Foundation était pour moi un sujet sur lequel j'avais déjà quelques connaissances.

En effet durant mon projet long de 3ème année, j'ai eu l'opportunité de travailler sur WCF.

Étant donné que le travail de diplôme est réalisé dans une durée de 3 mois, les objectifs étaient autres.

Il était clair, que ce travail allait être intéressant. Et travailler avec de nouvelles technologies est toujours très enrichissant. Malheureusement il n'est jamais facile d'utiliser des technologies qui sont en développement. Pour cela j'imaginais également, que de nombreux challenges allaient intervenir tout au long de mon projet.

2. Planification et suivi du travail de diplôme

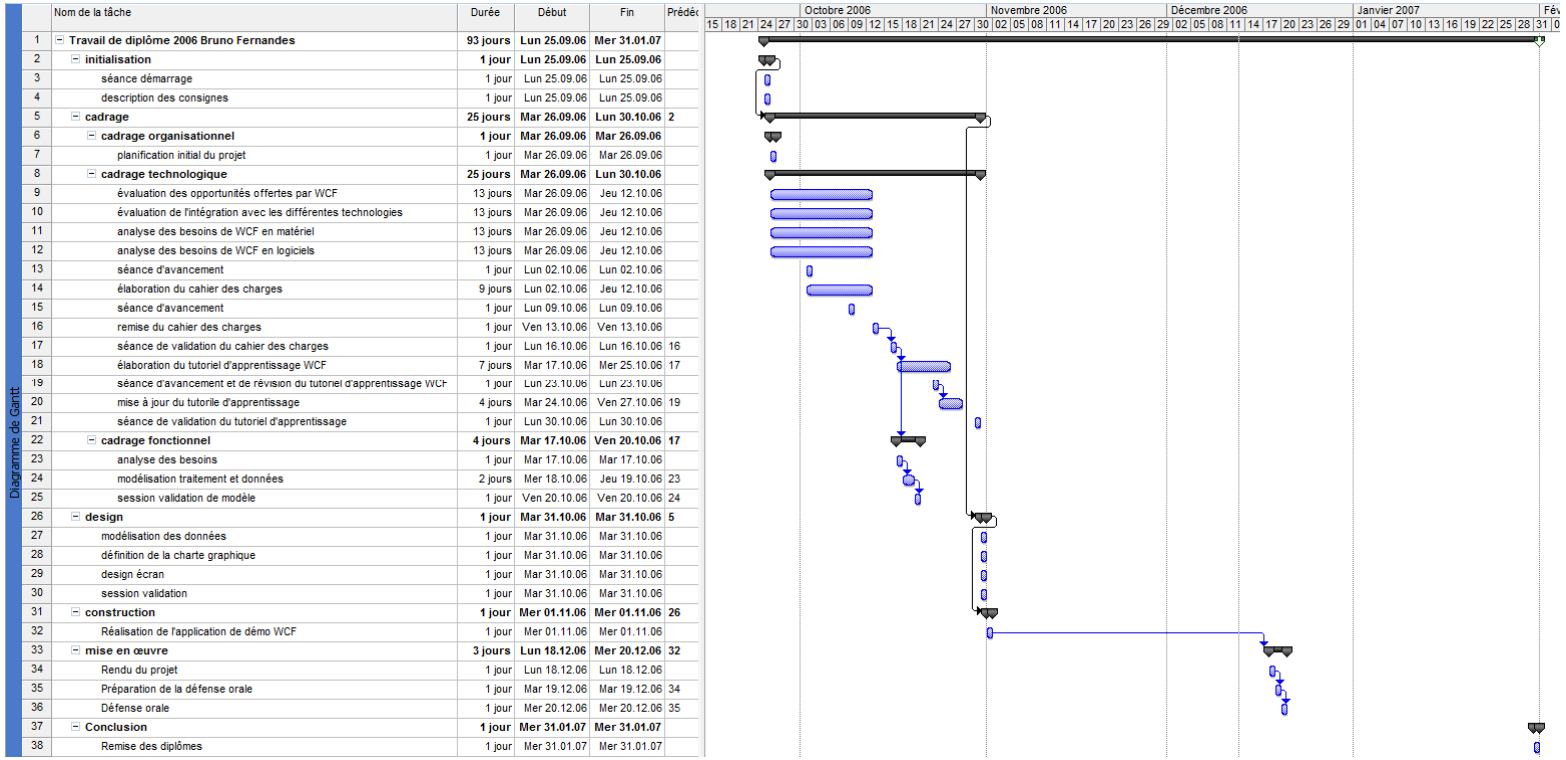
2.1. Planification du Projet

La première approche de mon travail de diplôme a été de réaliser une planification du travail à l'aide de Microsoft Projet.

Etant la seule personne travaillant sur ce projet, l'affectation des ressources dans MS Projet montrait peu d'intérêt. Cette planification a donc été utilisée pour fixer des échéances et avoir une vue globale.

Vous trouverez la planification initiale et finale au chapitre 3.1 et 3.2. Toutefois, le développement concret n'étant pas encore fixé lors de la réalisation de la planification initiale, les tâches s'y rapportant n'ont donc pas été planifiées.

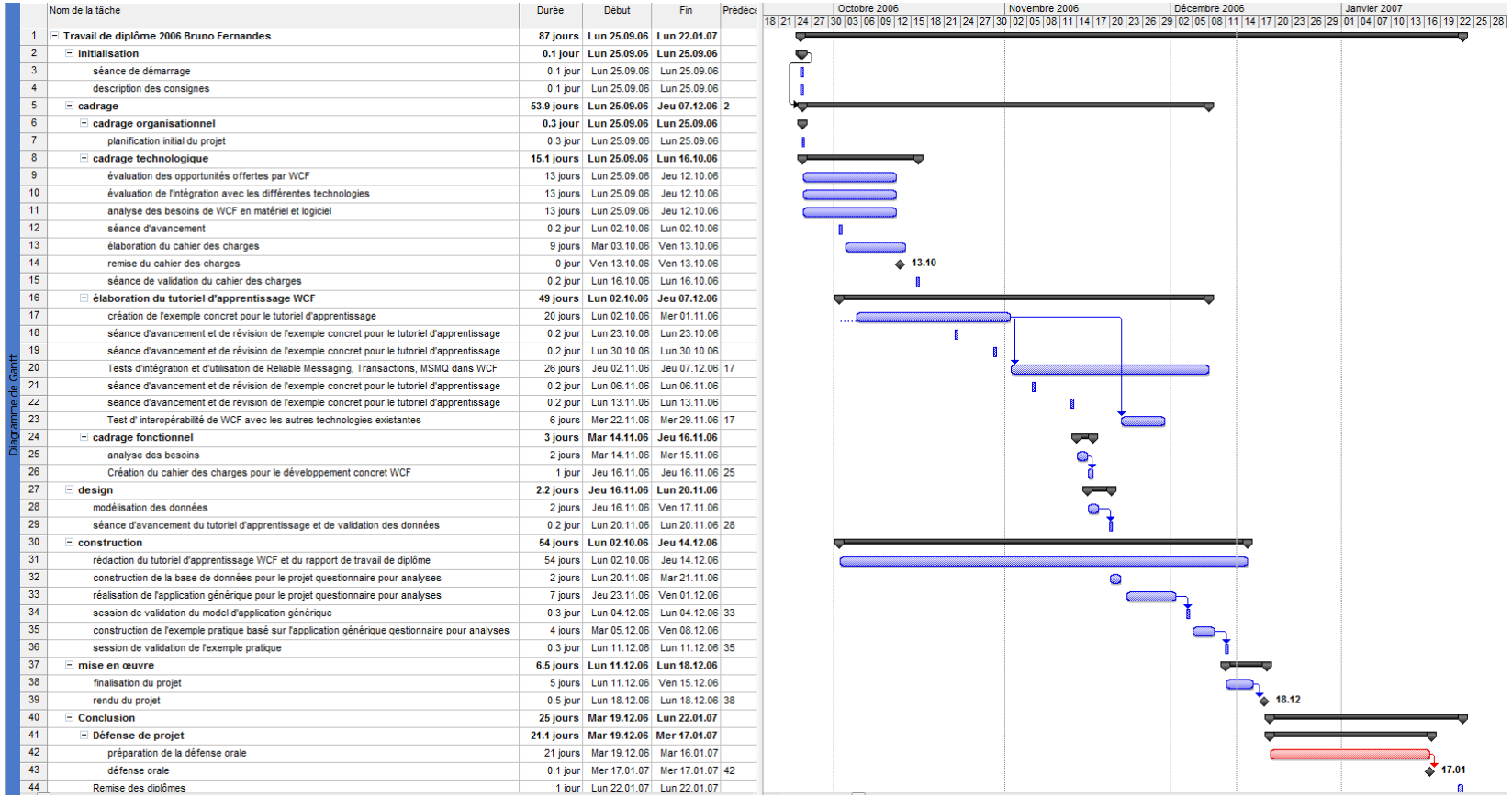
Planification initiale



	Nom de la tâche	Durée	Début	Fin	Prédéc
1	**Travail de diplôme 2006 Bruno Fernandes**	**93 jours**	**Lun 25.09.06**	**Mer 31.01.07**	
2	**initialisation**	**1 jour**	**Lun 25.09.06**	**Lun 25.09.06**	
3	séance démarrage	1 jour	Lun 25.09.06	Lun 25.09.06	
4	description des consignes	1 jour	Lun 25.09.06	Lun 25.09.06	
5	**cadrage**	**25 jours**	**Mar 26.09.06**	**Lun 30.10.06**	**2**
6	**cadrage organisationnel**	**1 jour**	**Mar 26.09.06**	**Mar 26.09.06**	
7	planification initial du projet	1 jour	Mar 26.09.06	Mar 26.09.06	
8	**cadrage technologique**	**25 jours**	**Mar 26.09.06**	**Lun 30.10.06**	
9	évaluation des opportunités offertes par WCF	13 jours	Mar 26.09.06	Jeu 12.10.06	
10	évaluation de l'intégration avec les différentes technologies	13 jours	Mar 26.09.06	Jeu 12.10.06	
11	analyse des besoins de WCF en matériel	13 jours	Mar 26.09.06	Jeu 12.10.06	
12	analyse des besoins de WCF en logiciels	13 jours	Mar 26.09.06	Jeu 12.10.06	
13	séance d'avancement	1 jour	Lun 02.10.06	Lun 02.10.06	
14	élaboration du cahier des charges	9 jours	Lun 02.10.06	Jeu 12.10.06	
15	séance d'avancement	1 jour	Lun 09.10.06	Lun 09.10.06	
16	remise du cahier des charges	1 jour	Ven 13.10.06	Ven 13.10.06	
17	séance de validation du cahier des charges	1 jour	Lun 16.10.06	Lun 16.10.06	16
18	élaboration du tutoriel d'apprentissage WCF	7 jours	Mar 17.10.06	Mer 25.10.06	17
19	séance d'avancement et de révision du tutoriel d'apprentissage WCF	1 jour	Lun 23.10.06	Lun 23.10.06	
20	mise à jour du tutorile d'apprentissage	4 jours	Mar 24.10.06	Ven 27.10.06	19
21	séance de validation du tutoriel d'apprentissage	1 jour	Lun 30.10.06	Lun 30.10.06	
22	**cadrage fonctionnel**	**4 jours**	**Mar 17.10.06**	**Ven 20.10.06**	**17**
23	analyse des besoins	1 jour	Mar 17.10.06	Mar 17.10.06	
24	modélisation traitement et données	2 jours	Mer 18.10.06	Jeu 19.10.06	23
25	session validation de modèle	1 jour	Ven 20.10.06	Ven 20.10.06	24
26	**design**	**1 jour**	**Mar 31.10.06**	**Mar 31.10.06**	**5**
27	modélisation des données	1 jour	Mar 31.10.06	Mar 31.10.06	
28	définition de la charte graphique	1 jour	Mar 31.10.06	Mar 31.10.06	
29	design écran	1 jour	Mar 31.10.06	Mar 31.10.06	
30	session validation	1 jour	Mar 31.10.06	Mar 31.10.06	
31	**construction**	**1 jour**	**Mer 01.11.06**	**Mer 01.11.06**	**26**
32	Réalisation de l'application de démo WCF	1 jour	Mer 01.11.06	Mer 01.11.06	
33	**mise en œuvre**	**3 jours**	**Lun 18.12.06**	**Mer 20.12.06**	**32**
34	Rendu du projet	1 jour	Lun 18.12.06	Lun 18.12.06	
35	Préparation de la défense orale	1 jour	Mar 19.12.06	Mar 19.12.06	34
36	Défense orale	1 jour	Mer 20.12.06	Mer 20.12.06	35
37	**Conclusion**	**1 jour**	**Mer 31.01.07**	**Mer 31.01.07**	
38	Remise des diplômes	1 jour	Mer 31.01.07	Mer 31.01.07	

Diagramme de Gantt de la planification initiale créée avec Ms Projet.

Planification finale



	Nom de la tâche	Durée	Début	Fin	Prédéc
1	**Travail de diplôme 2006 Bruno Fernandes**	**87 jours**	**Lun 25.09.06**	**Lun 22.01.07**	
2	**initialisation**	**0.1 jour**	**Lun 25.09.06**	**Lun 25.09.06**	
3	séance de démarrage	0.1 jour	Lun 25.09.06	Lun 25.09.06	
4	description des consignes	0.1 jour	Lun 25.09.06	Lun 25.09.06	
5	**cadrage**	**53.9 jours**	**Lun 25.09.06**	**Jeu 07.12.06**	2
6	**cadrage organisationnel**	**0.3 jour**	**Lun 25.09.06**	**Lun 25.09.06**	
7	planification initial du projet	0.3 jour	Lun 25.09.06	Lun 25.09.06	
8	**cadrage technologique**	**15.1 jours**	**Lun 25.09.06**	**Lun 16.10.06**	
9	évaluation des opportunités offertes par WCF	13 jours	Lun 25.09.06	Jeu 12.10.06	
10	évaluation de l'intégration avec les différentes technologies	13 jours	Lun 25.09.06	Jeu 12.10.06	
11	analyse des besoins de WCF en matériel et logiciel	13 jours	Lun 25.09.06	Jeu 12.10.06	
12	séance d'avancement	0.2 jour	Lun 02.10.06	Lun 02.10.06	
13	élaboration du cahier des charges	9 jours	Mar 03.10.06	Ven 13.10.06	
14	remise du cahier des charges	0 jour	Ven 13.10.06	Ven 13.10.06	
15	séance de validation du cahier des charges	0.2 jour	Lun 16.10.06	Lun 16.10.06	
16	**élaboration du tutoriel d'apprentissage WCF**	**49 jours**	**Lun 02.10.06**	**Jeu 07.12.06**	
17	création de l'exemple concret pour le tutoriel d'apprentissage	20 jours	Lun 02.10.06	Mer 01.11.06	
18	séance d'avancement et de révision de l'exemple concret pour le tutoriel d'apprentissage	0.2 jour	Lun 23.10.06	Lun 23.10.06	
19	séance d'avancement et de révision de l'exemple concret pour le tutoriel d'apprentissage	0.2 jour	Lun 30.10.06	Lun 30.10.06	
20	Tests d'intégration et d'utilisation de Reliable Messaging, Transactions, MSMQ dans WCF	26 jours	Jeu 02.11.06	Jeu 07.12.06	17
21	séance d'avancement et de révision de l'exemple concret pour le tutoriel d'apprentissage	0.2 jour	Lun 06.11.06	Lun 06.11.06	
22	séance d'avancement et de révision de l'exemple concret pour le tutoriel d'apprentissage	0.2 jour	Lun 13.11.06	Lun 13.11.06	
23	Test d' interopérabilité de WCF avec les autres technologies existantes	6 jours	Mer 22.11.06	Mer 29.11.06	17
24	**cadrage fonctionnel**	**3 jours**	**Mar 14.11.06**	**Jeu 16.11.06**	
25	analyse des besoins	2 jours	Mar 14.11.06	Mer 15.11.06	
26	Création du cahier des charges pour le développement concret WCF	1 jour	Jeu 16.11.06	Jeu 16.11.06	25
27	**design**	**2.2 jours**	**Jeu 16.11.06**	**Lun 20.11.06**	
28	modélisation des données	2 jours	Jeu 16.11.06	Ven 17.11.06	
29	séance d'avancement du tutoriel d'apprentissage et de validation des données	0.2 jour	Lun 20.11.06	Lun 20.11.06	28
30	**construction**	**54 jours**	**Lun 02.10.06**	**Jeu 14.12.06**	
31	rédaction du tutoriel d'apprentissage WCF et du rapport de travail de diplôme	54 jours	Lun 02.10.06	Jeu 14.12.06	
32	construction de la base de données pour le projet questionnaire pour analyses	2 jours	Lun 20.11.06	Mar 21.11.06	
33	réalisation de l'application générique pour le projet questionnaire pour analyses	7 jours	Jeu 23.11.06	Ven 01.12.06	
34	session de validation du model d'application générique	0.3 jour	Lun 04.12.06	Lun 04.12.06	33
35	construction de l'exemple pratique basé sur l'application générique qestionnaire pour analyses	4 jours	Mar 05.12.06	Ven 08.12.06	
36	session de validation de l'exemple pratique	0.3 jour	Lun 11.12.06	Lun 11.12.06	35
37	**mise en œuvre**	**6.5 jours**	**Lun 11.12.06**	**Lun 18.12.06**	
38	finalisation du projet	5 jours	Lun 11.12.06	Ven 15.12.06	
39	rendu du projet	0.5 jour	Lun 18.12.06	Lun 18.12.06	38
40	**Conclusion**	**25 jours**	**Mar 19.12.06**	**Lun 22.01.07**	
41	**Défense de projet**	**21.1 jours**	**Mar 19.12.06**	**Mer 17.01.07**	
42	préparation de la défense orale	21 jours	Mar 19.12.06	Mar 16.01.07	
43	défense orale	0.1 jour	Mer 17.01.07	Mer 17.01.07	42
44	Remise des diplômes	1 jour	Lun 22.01.07	Lun 22.01.07	

Diagramme de Gantt de la planification finale créée avec Ms Projet.

Il a été difficile de respecter la planification initialement prévue.

Plusieurs raisons expliquent cela :

- Technologie en développement

  Au début de mon travail le .Net Framework 3.0 était encore en version RC (Release candidate) et a évolué en version finale en espace de 2 mois.

  Pour pouvoir rendre un tutoriel complet et d'actualité, je devais suivre l'évolution de cette technologie. Ainsi mon tutoriel est basé sur une version finale du .Net Framework 3.0.

  Ceci a mené à plusieurs contraintes se situant à différents niveaux:

  - Matériel :
    - la configuration des machines à du être refaite. Réinstallation du Framework, Windows SDK et extensions pour le Framework.
  - Logiciels :
    - les exemples ont du à nouveau être testés. Ceci, dans le but de s'assurer que leur code soit toujours d'actualité avec le Framework.
  - Tutoriel :
    - Plusieurs chapitres du tutoriel ont du être à nouveau écrits ou modifiés.
- Interopérabilité avec de nombreuses technologies inconnues

  Un des objectifs du tutoriel était de démontrer l'interopérabilité avec les technologies existantes. La plupart de ces technologies étaient pour moins, inconnues ou peu maitrisées.

  Il était donc difficile, de déterminer le temps qu'il fallait mettre à disposition pour leur réalisation.
- Sources inexactes ou inexistantes

  Il existe plusieurs exemple ou sources qui malheureusement ne sont plus à jour. En effet WCF a beaucoup évolué et plusieurs méthodes ont été remplacées ou n'existent tout simplement plus.

  WCF permet la paramétrisation très complète de la couche communication d'un web service. Ceci a des avantages énormes, mais augmente également la difficulté à régler de manière optimum la communication.

Du à cela, j'ai rencontré un problème dans mon exemple « BankingService » du tutoriel WCF. Cette complication m'a fait perdre de nombreux jours (Temps approximativement estimé à une semaine).

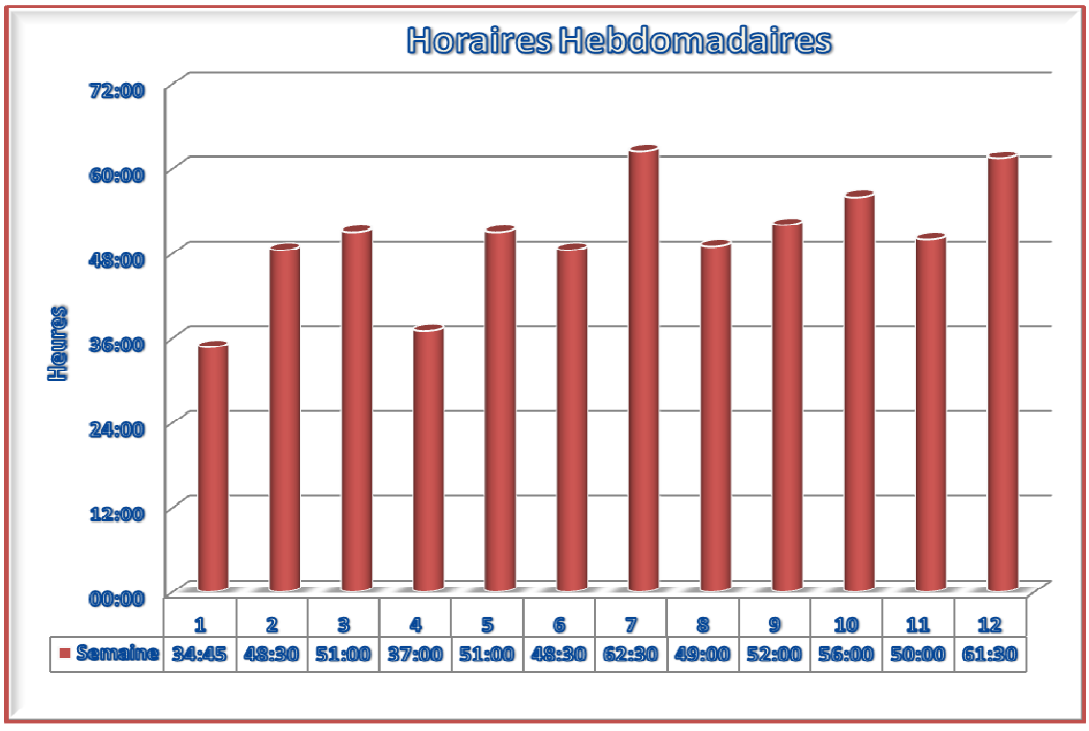
Même en faisait appel à l'aide des formeurs du site officiel de WCF, je n'ai pas réussi à trouver de solution.

Finalement à force de chercher et d'effectuer des tests, j'ai découvert qu'un simple paramètre du fichier de configuration devait être modifié.

2.2. Travail hebdomadaire et suivi de projet

La charge de travail hebdomadaire a été calculée à une moyenne de 50 heures et 10 minutes.

Voici un graphique de la répartition des heureux durant le projet :



Horaires de travail hebdomadaires

Afin de maintenir un bon suivi de projet et garantir un bon niveau d'avancement, j'ai réalisé pour toutes les semaines un rapport hebdomadaire.

Ses rapports contenaient :

- Les activités de la semaine ainsi que le temps mis en œuvre pour l'exécution de chacune.
- Les problèmes rencontrés ainsi que leur solution
- Les problèmes en cours (qui non pas encore de solution) ainsi que leur solution potentielle.
- La description du travail hebdomadaire ainsi que les objectifs pour la semaine à venir.

Ces rapports ont été transmis à mon responsable de travail de diplôme à la fin de chaque semaine.

Tous les lundis avait lieu une séance d'avancement, nous y discutions du travail hebdomadaire, de l'état d'avancement et des objectifs pour la semaine.

Vous trouverez sur le cd de ressources WCF ces rapports.

Répertoire : GestionProjet\RapportsHebdos

3. Déroulement du travail de diplôme

Le déroulement de mon travail s'est clairement divisé en deux parties distinctes qui sont:

- La réalisation d'un tutoriel d'apprentissage
- La réalisation d'un développement concret avec WCF

Dû a la phase d'apprentissage et d'analyse, la réalisation du tutoriel a été la partie qui a requis le plus de temps. Ensuite est venu la réalisation du développement concret.

Le tutoriel ayant demandé plus de temps que celui planifié, il a été terminé en parallèle au développement concret.

Vous trouverez dans le chapitre 5 et 6 une plus grande description de la réalisation de ces étapes.

Mais avant la réalisation de chacune, la première approche a été de créer un cahier des charges pour chaque étape afin d'établir clairement les objectifs et contraintes.

Vous trouverez ces cahiers des charges dans les annexes au chapitre 9.

4. Réalisation du Tutoriel sur l'utilisation de WCF

Cette étape a commencé par une Analyse complète des possibilités offertes par WCF. Ceci a permis de déterminer :

- les besoins à couvrir
- les contraintes liées à WCF et la réalisation du projet.
- les chapitres qui allaient être traitées dans le tutoriel.

La structure de mon travail étant mise en place, il me restait à rédiger ces chapitres et à construire les différents exemples qui allaient faire partie de mon tutoriel.

Pour des causes de réutilisabilité, le tutoriel ne fait pas partie intégrante de se rapport et se trouve en annexe au chapitre 9.

Je vous propose donc, de lire ce tutoriel pour vous faire une idée de l'étendue des fonctionnalités que WCF apporte.

5. Réalisation d'un développement concret WCF

A la remise du sujet de mon travail de diplôme, aucun thème pour le développement concret d'une application n'a été définit. En effet la thématique de ce travail allait être prise en cours de projet, selon les différents contacts industriels tels que Nestlé.

Malheureusement, aucun débouché pour la réalisation de ce travail dans un milieu industriel n'a pu aboutir.

Jean-Pierre Rey m'a alors proposé un thème, pour la réalisation de mon développement concret WCf.

5.1. Présentation du projet

Sujet du projet

L'idéale du projet serait de créer un web service qui pourrait servir de base à un générateur.

Dû à des contraintes de temps (application réalisable en 160h), le cahier des charges pour ce développement a été limité.

L'objectif est de réaliser une application générique à plusieurs analyses qui devrait se présenter sous forme de questionnaires.

Après avoir atteint ce premier objectif, une implémentation concrète basée sur cette application devra être mis en œuvre.

Cette implémentation aura une application cliente non générique qui va se charger de consommer le web service pour enregistrer le résultat d'une analyse.

En fonction d'un module métier, spécifique à chaque questionnaire, un résultat sera alors généré.

Délivrable du projet

Voici une liste exhaustive du travail réalisé pour l'application générique:

- Scriptes pour la génération de la base de données Microsoft SQL Server et procédures stockées

- Web Service WCF
- Interface pour l'accès au web service pour le client

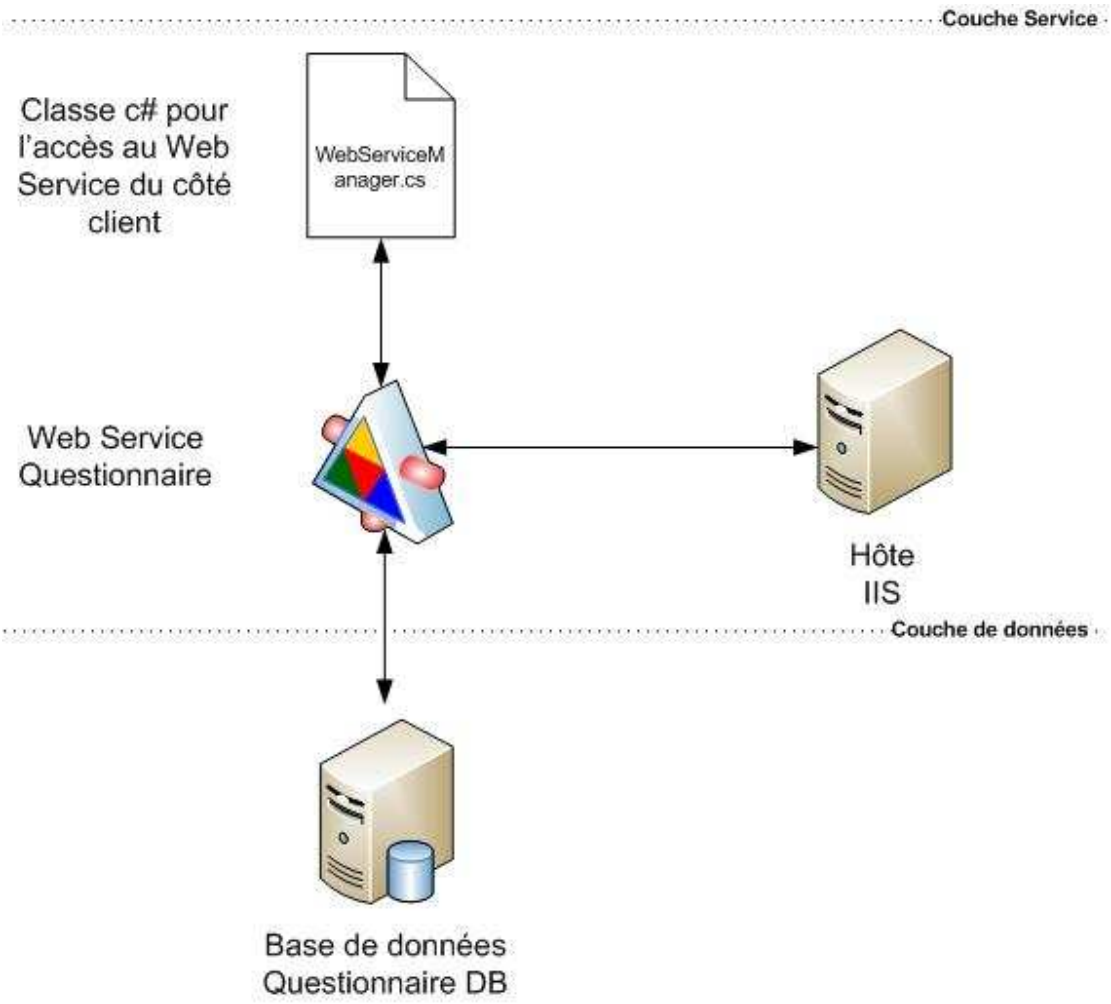


Schéma du Projet Questionnaires (générique)

Vous remarquez sur ce schéma que:

- Le web service sera relié à une base de données sur laquelle seront stockées les données de l'application.
- Le web service sera hébergé par IIS.

Après avoir développé le projet générique, il faut implémenter un exemple concret basé sur ce dernier.

Voici une liste exhaustive du travail réalisé :

- Scriptes pour la génération des données de l'exemple
- Module métier qui va calculer un profile en fonction d'une analyse
- Application cliente qui va se charger du déroulement de l'analyse.

IL est à noter, que ces éléments ne sont pas génériques. Par conséquent, à chaque implémentation du projet générique, ils seront à réécrire ou à modifier.

5.2. Objectifs et approche vis-à-vis du projet

L'objectif de ce projet était clairement de réaliser un web service WCF réutilisable et de prouver son utilisation avec un exemple concret.

Mon approche a donc été de consulter le maximum d'analyses et de questionnaires disponibles sur la toile.

Ainsi, j'ai pu avoir une vison très large des différentes structures de questionnaires.

5.3. Modélisation des données

Après la première approche d'analyse, est venue la modélisation de la base de données. Celle-ci a été faite à l'aide de l'outil Power Designer.

Voici la structure de la base de données:

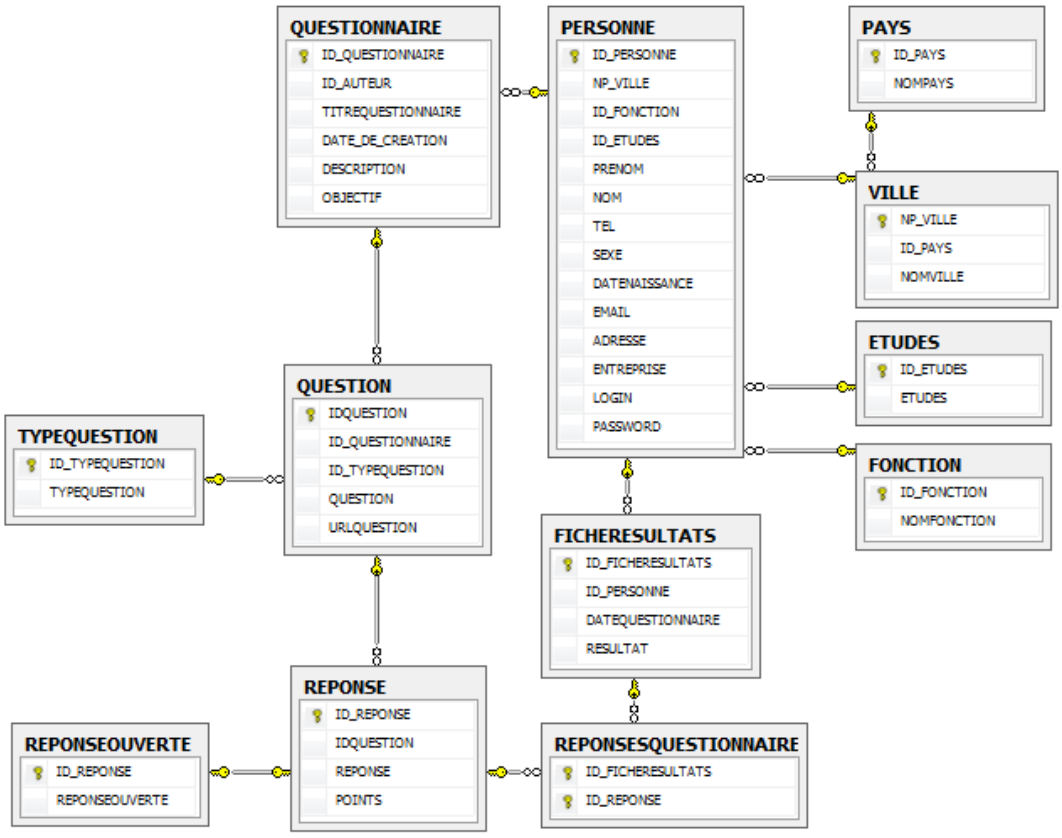


Diagramme de la base de données QuestionnaireDB

La base de données comprend les tables suivantes :

- Personne

  Cette table stocke les données d'un utilisateur. Elle permet d'amasser une grande quantité de données personnelles. Celles-ci pourront être utilisées par les modules métiers.

- Ville

Cette table stocke les données des villes.

- Pays

  Cette table stocke les données des pays.

- Etudes

  Cette table stocke les données des études. Ces données peuvent être par exemple : «HES », « EPFL »,…

- Fonction

  Cette table stocke les données des fonctions d'une personne. Ces données peuvent être par exemple : « Informaticien », « Employé de commerce », …

- Questionnaire

  Cette table stocke les données des questionnaires. Un questionnaire est créé pour une personne, a un titre et une date de création. Il est également possible d'y ajouter une description et des objectifs.

- Question

  Cette table stocke les questions. Une question est associée à un questionnaire et a un type. Elle peut également avoir une url, pour une source externe, telle qu'une image.

- TypeQuestion

  Cette table stocke les types de questions. Un type de question est par exemple : QCM, Oui-Non, Ouverte, …

- Reponse

  Cette table stocke les réponses. Une réponse est associée à une question. Les champs points défini les points de la question. Selon ces points le module métier pourra alors calculer un profile.

- ReponseOuverte

  Cette table stocke les compléments à une réponse. Un complément, représente donc une réponse ouverte.

- ReponsesQuestionnaires

  Cette table stocke les résultats des analyses. Elle enregistre les réponses choisies pour une fichier de résultats.

- FichierResultats

Cette table stocke la fiche de résultats de l’analyse. Une fiche, a un résultat pour une personne, ayant fait un questionnaire à une date précise.

Vous trouverez les scriptes pour la création de la base de données et procédures stockées dans le cd de ressources WCF.

Répertoire: ProjetQuestionnaire\ProjetGénérique\ScriptsSQL

5.4. Web Service Questionnaire

Avant d'aborder les chapitres à venir, il est important d'avoir de bonnes connaissances sur la structure d'un web service WCF. Pour cette raison, il est requis de lire le tutoriel WCF annexé.

Le projet « QuestionnaireService » est composé de plusieurs classes. Nous allons voir leur structure et leur utilité.

Pour avoir une version informatique des ce projet, veuillez consulter le cd de ressources WCF.

Répertoire:
ProjetQuestionnaire\ProjetGénérique\QuestionnaireServiceSolution

Contract

Le contract défini plusieurs méthodes qui vont permettre l'interaction avec la base de données QuestionnaireDB.

Voici le code du contract:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.ServiceModel;
using System.Runtime.Serialization;
using System.Data;

namespace QuestionnaireService
{
    /// <summary>
    /// Interface QuestionnaireContract qui défini les méthodes du web service qui seront exposées
    /// </summary>
    [ServiceContract()]
    public interface QuestionnaireContract
    {
        [OperationContract]
        [FaultContract(typeof(QuestionnaireServiceException))]
        DataSet getAllPersonnes();

        [OperationContract]
        [FaultContract(typeof(QuestionnaireServiceException))]
        DataSet getPersonne(string login, string psw);

        [OperationContract]
        [FaultContract(typeof(QuestionnaireServiceException))]
        void insertEtudes(string etudes);

        [OperationContract]
        [FaultContract(typeof(QuestionnaireServiceException))]
        void insertResultat(int idFicheResultat, string resultat);
```

```
        [OperationContract]
        [FaultContract(typeof(QuestionnaireServiceException))]
        DataSet getFichesResultatsPersonne(int idPersonne);

        [OperationContract]
        [FaultContract(typeof(QuestionnaireServiceException))]
        DataSet getFicheResultats(int idFicheResultat);

        [OperationContract]
        [FaultContract(typeof(QuestionnaireServiceException))]
        int insertFicheResultats(int idPersonne);

        [OperationContract]
        [FaultContract(typeof(QuestionnaireServiceException))]
        void insertFonction(string nomFonction);

        [OperationContract]
        [FaultContract(typeof(QuestionnaireServiceException))]
        void insertPays(string nomPays);

        [OperationContract]
        [FaultContract(typeof(QuestionnaireServiceException))]
        void insertVille(int idPays, int npVille, string nomVille);

        [OperationContract]
        [FaultContract(typeof(QuestionnaireServiceException))]
        void insertTypeQuestion(string typeQuestion);

        [OperationContract]
        [FaultContract(typeof(QuestionnaireServiceException))]
        DataSet getPersonneID(int idPersonne);

        [OperationContract]
        [FaultContract(typeof(QuestionnaireServiceException))]
        void insertPersonne(int idFonction, int idEtudes, string prenom, string nom, string sexe,
string dateNaissance, string tel, string email, string adresse, int npVille, string entreprise, string
login, string password);

        [OperationContract]
        [FaultContract(typeof(QuestionnaireServiceException))]
        void insertPersonneObject(Personne pers);

        [OperationContract]
        [FaultContract(typeof(QuestionnaireServiceException))]
        DataSet getQuestionnaires();

        [OperationContract]
        [FaultContract(typeof(QuestionnaireServiceException))]
        void insertQuestionnaire(int idAuteur, string titreQuestionnaire, string description, string
objectif);

        [OperationContract]
        [FaultContract(typeof(QuestionnaireServiceException))]
        void insertQuestionnaireObject(Questionnaire quest);

        [OperationContract]
        [FaultContract(typeof(QuestionnaireServiceException))]
        DataSet getQuestions(int idQuestionnaire);

        [OperationContract]
        [FaultContract(typeof(QuestionnaireServiceException))]
```

```
        void insertQuestion(int idQuestionnaire, int idTypeQuestion, string Question, string
urlQuestion);

        [OperationContract]
        [FaultContract(typeof(QuestionnaireServiceException))]
        void insertQuestionObject(Question quest);

        [OperationContract]
        [FaultContract(typeof(QuestionnaireServiceException))]
        DataSet getReponses(int idQuestion);

        [OperationContract]
        [FaultContract(typeof(QuestionnaireServiceException))]
        int getPointReponse(int idReponse);

        [OperationContract]
        [FaultContract(typeof(QuestionnaireServiceException))]
        void insertReponse(int idQuestion, string reponse, int points);

        [OperationContract]
        [FaultContract(typeof(QuestionnaireServiceException))]
        void insertReponseObject(Reponse r);

        [OperationContract]
        [FaultContract(typeof(QuestionnaireServiceException))]
        void insertReponseOuverte(int idReponse, string ReponseOuverte);

        [OperationContract]
        [FaultContract(typeof(QuestionnaireServiceException))]
        void insertReponseOuverteObject(ReponseOuverte r);

        [OperationContract]
        [FaultContract(typeof(QuestionnaireServiceException))]
        void insertReponseQuestionnaire(int idFicheResultats, int idReponse);

    }

    /// <summary>
    /// Classe permettant de gérer les exceptions soap dans notre web service
    /// </summary>
    [DataContract]
    public class QuestionnaireServiceException
    {
        [DataMember]
        public string title;
        [DataMember]
        public string text;
        [DataMember]
        public string details;
    }

    /// <summary>
    /// classe personne permettant de sérialiser un object de type personne
    /// </summary>
    [DataContract]
    public class Personne
    {
        [DataMember]
        public int idFonction;
        [DataMember]
        public int idEtudes;
        [DataMember]
```

```
    public string prenom;
    [DataMember]
    public string nom;
    [DataMember]
    public string sexe;
    [DataMember]
    public string dateNaissance;
    [DataMember]
    public string tel;
    [DataMember]
    public string email;
    [DataMember]
    public string adresse;
    [DataMember]
    public int npVille;
    [DataMember]
    public string entreprise;
    [DataMember]
    public string login;
    [DataMember]
    public string password ;
}

/// <summary>
/// Object Questionnaire
/// </summary>
[DataContract]
public class Questionnaire
{
    [DataMember]
    public int idAuteur;
    [DataMember]
    public string titreQuestionnaire;
    [DataMember]
    public string description;
    [DataMember]
    public string objectifs ;
}

/// <summary>
/// Object Question
/// </summary>
[DataContract]
public class Question
{
    [DataMember]
    public int idQuestionnaire;
    [DataMember]
    public int idTypeQuestion;
    [DataMember]
    public string question;
    [DataMember]
    public string urlQuestion;
}

/// <summary>
/// Object Reponse
/// </summary>
[DataContract]
public class Reponse
{
    [DataMember]
```

```
        public int idQuestion;
        [DataMember]
        public string reponse;
        [DataMember]
        public int points;
    }

    /// <summary>
    /// Objet ReponseOuverte
    /// </summary>
    [DataContract]
    public class ReponseOuverte
    {
        [DataMember]
        public int idReponse;
        [DataMember]
        public string reponseOuverte;
    }
}
```

Avec les attributs [DataContract] & [DataMember] les objets suivants pourront être utilisés avec notre Web Service :

- QuestionnaireServiceException
- Personne
- Questionnaire
- Question
- Reponse
- ReponseOuverte

La gestion des exception est faite avec la classe générique FaultException et l'objet « QuestionnaireServiceException ».

ContractType

Le ContractType est une implémentation de l' l'interface QuestonnaireContract.

Voici son code :

```
using System;
using System.Collections.Generic;
using System.ServiceModel;
using System.Text;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;

namespace QuestionnaireService
{
   class QuestionnaireContractType : QuestionnaireContract
   {
      /// <summary>
      /// variables
      /// </summary>
      DataSet ds;
      string procName;
      string errorTxt;
      DbConnectionManager con;

      /// <summary>
      /// Méhtode qui va récupérer toutes les personnes de la base de données
      /// </summary>
      /// <returns>retourne un dataset contenant toutes les personnes</returns>
      DataSet QuestionnaireContract.getAllPersonnes()
      {
         ds = new DataSet();
         procName = "sp_selectAllPersonnes";
         errorTxt = "Récupération de tous les clients" ;
         con = new DbConnectionManager();

         try
         {
            ds=con.selectProcWithoutParam (procName);
         }
         catch (Exception ex)
         {
            QuestionnaireServiceException error = new QuestionnaireServiceException();
            error.title = "Erreur QuestionnaireService : " + errorTxt;
            error.text = error.title;
            error.details = "\n Procédure Stockée : "+ procName +" \n " + ex.Message;
            throw new FaultException<QuestionnaireServiceException>(error);
         }

         return ds;
      }


      /// <summary>
      /// Méthode qui va récupér une personne en fonction de son login et mot de passe
      /// </summary>
      /// <param name="login">login de la personne</param>
      /// <param name="psw">mot de passe de la personne</param>
```

```
/// <returns>un dataset contenant la personne récupérée</returns>
DataSet QuestionnaireContract.getPersonne(string login, string psw)
{
    ds = new DataSet();
    procName = "sp_selectPersonne";
    errorTxt = "Récupération d'un client avec login et mo de passe";
    con = new DbConnectionManager();

    try
    {
        //définition de la procédure stockée
        con.selectProc(procName);

        //Ajout de paramètres
        con.Com.Parameters.Add("@Login", SqlDbType.VarChar).Value = login;
        con.Com.Parameters.Add("@Psq", SqlDbType.VarChar).Value = psw;
        ds = con.fillDataset();
    }
    catch (Exception ex)
    {
        QuestionnaireServiceException error = new QuestionnaireServiceException();
        error.title = "Erreur QuestionnaireService : " + errorTxt;
        error.text = error.title + "login = " + login;
        error.details = "\n Procédure Stockée : " + procName + " \n " + ex.Message;
        throw new FaultException<QuestionnaireServiceException>(error);
    }

    return ds;
}

/// <summary>
/// Méthode qui va récupérer toutes le fiches de résultat d'une personne
/// </summary>
/// <param name="idPersonne">identifiant de la personne</param>
/// <returns>retourne un dataset avec les fiches de résultats</returns>
DataSet QuestionnaireContract.getFichesResultatsPersonne(int idPersonne)
{
    ds = new DataSet();
    procName = "sp_selectFichesResultatsPersonne";
    errorTxt = "Récupération des fiches de résultats selon un identifiant d'une personne";
    con = new DbConnectionManager();

    try
    {
        //définition de la procédure stockée
        con.selectProc(procName);

        //Ajout de paramètres
        con.Com.Parameters.Add("@IDPERSONNE", SqlDbType.Int).Value = idPersonne;
        ds = con.fillDataset();
    }
    catch (Exception ex)
    {
        QuestionnaireServiceException error = new QuestionnaireServiceException();
        error.title = "Erreur QuestionnaireService : " + errorTxt;
        error.text = error.title + " idPersonne = " + idPersonne;
        error.details = "\n Procédure Stockée : " + procName + " \n " + ex.Message;
        throw new FaultException<QuestionnaireServiceException>(error);
    }

    return ds;
}
```

```
        /// <summary>
        /// Méthode qui va récupérer une fiche de résultats
        /// </summary>
        /// <param name="idFicheResultat">identifiant de la fiche de résultars</param>
        /// <returns>dataset contenant la fiche de résultat de la personne</returns>
        DataSet QuestionnaireContract.getFicheResultats(int idFicheResultat)
        {
            ds = new DataSet();
            procName = "sp_selectFicheResultats";
            errorTxt = "Récupération d'une fiche de résultats selon son identifiant";
            con = new DbConnectionManager();

            try
            {
                //définition de la procédure stockée
                con.selectProc(procName);

                //Ajout de paramètres
                con.Com.Parameters.Add("@ID_FICHERESULTATS", SqlDbType.Int).Value =
idFicheResultat;
                ds = con.fillDataset();
            }
            catch (Exception ex)
            {
                QuestionnaireServiceException error = new QuestionnaireServiceException();
                error.title = "Erreur QuestionnaireService : " + errorTxt;
                error.text = error.title + " idFicheResultat = " + idFicheResultat;
                error.details = "\n Procédure Stockée : " + procName + " \n " + ex.Message;
                throw new FaultException<QuestionnaireServiceException>(error);
            }

            return ds;
        }

        /// <summary>
        /// Méthode qui va insérer le résultat d'un questionnaire à une fiche de résultat
        /// </summary>
        /// <param name="idFicheResultat">identifiant de la fiche de résultas</param>
        /// <param name="resultat">résultat</param>
        void QuestionnaireContract.insertResultat(int idFicheResultat, string resultat)
        {
            procName = "sp_insertResultat";
            errorTxt = "Insertion d'un Résultat";
            con = new DbConnectionManager();

            try
            {
                //définition de la procédure stockée
                con.useProc(procName);

                //Ajout de paramètres
                con.Com.Parameters.Add("@ID_FICHERESULTATS", SqlDbType.Int).Value =
idFicheResultat;
                con.Com.Parameters.Add("@RESULTAT", SqlDbType.Text).Value = resultat;

                con.execProc();
            }
            catch (Exception ex)
            {
                QuestionnaireServiceException error = new QuestionnaireServiceException();
```

```
            error.title = "Erreur QuestionnaireService : " + errorTxt;
            error.text = error.title + " idFicheResultat = " + idFicheResultat + " resultat = " + resultat;
            error.details = "\n Procédure Stockée : " + procName + " \n " + ex.Message;
            throw new FaultException<QuestionnaireServiceException>(error);
        }
    }


    /// <summary>
    /// Méthode qui insérer un type d'étude dans la base de données EX: HES
    /// </summary>
    /// <param name="etudes">nom de l'étude</param>
    void QuestionnaireContract.insertEtudes(string etudes)
    {
        procName = "sp_insertEtudes";
        errorTxt = "Insertion d'une Etude";
        con = new DbConnectionManager();

        try
        {
            //définition de la procédure stockée
            con.useProc(procName);

            //Ajout de paramètres
            con.Com.Parameters.Add("@ETUDES", SqlDbType.VarChar).Value = etudes ;

            //Exécution de la procédure
            con.execProc();
        }
        catch (Exception ex)
        {
            QuestionnaireServiceException error = new QuestionnaireServiceException();
            error.title = "Erreur QuestionnaireService : " + errorTxt;
            error.text = error.title + " nom de l'étude = " + etudes ;
            error.details = "\n Procédure Stockée : " + procName + " \n " + ex.Message;
            throw new FaultException<QuestionnaireServiceException>(error);
        }
    }

    /// <summary>
    /// Méthode qui va crée une fiche de résultat pour une personne
    /// </summary>
    /// <param name="idPersonne">identifiant de la personne</param>
    int QuestionnaireContract.insertFicheResultats(int idPersonne)
    {
        int idFicheResultat = -1;
        procName = "sp_insertFicheResultats";
        errorTxt = "Insertion d'une fiche de résultats";
        con = new DbConnectionManager();

        try
        {
            //définition de la procédure stockée
            con.useProc(procName);

            //Ajout de paramètres
            con.Com.Parameters.Add("@ID_PERSONNE", SqlDbType.Int).Value = idPersonne;

            //Execution de la procédure
            idFicheResultat = Convert.ToInt32(con.execScalar());

        }
```

```
            catch (Exception ex)
            {
                QuestionnaireServiceException error = new QuestionnaireServiceException();
                error.title = "Erreur QuestionnaireService : " + errorTxt;
                error.text = error.title + " idPersonne = " + idPersonne;
                error.details = "\n Procédure Stockée : " + procName + " \n " + ex.Message;
                throw new FaultException<QuestionnaireServiceException>(error);
            }
            return idFicheResultat;
        }

        /// <summary>
        /// Méthode qui va insérer une fonction dans la base de données EX: Informaticien
        /// </summary>
        /// <param name="nomFonction">nom de la fonction</param>
        void QuestionnaireContract.insertFonction(string nomFonction)
        {
            procName = "sp_insertFonction";
            errorTxt = "Insertion d'une fonction";
            con = new DbConnectionManager();

            try
            {
                //définition de la procédure stockée
                con.useProc(procName);

                //Ajout de paramètres
                con.Com.Parameters.Add("@NOMFONCTION", SqlDbType.VarChar).Value =
nomFonction;

                //Execution de la procédure
                con.execProc();

            }
            catch (Exception ex)
            {
                QuestionnaireServiceException error = new QuestionnaireServiceException();
                error.title = "Erreur QuestionnaireService : " + errorTxt;
                error.text = error.title + " nomFonction " + nomFonction;
                error.details = "\n Procédure Stockée : " + procName + " \n " + ex.Message;
                throw new FaultException<QuestionnaireServiceException>(error);
            }
        }

        /// <summary>
        /// Méthode qui va insérer un pays dans la base de données EX: Suisse
        /// </summary>
        /// <param name="nomPays">nom du pays</param>
        void QuestionnaireContract.insertPays(string nomPays)
        {
            procName = "sp_insertPays";
            errorTxt = "Insertion d'un Pays";
            con = new DbConnectionManager();

            try
            {
                //définition de la procédure stockée
                con.useProc(procName);

                //Ajout de paramètres
                con.Com.Parameters.Add("@NOMPAYS", SqlDbType.VarChar).Value = nomPays;
```

```
            //Execution de la procédure
            con.execProc();

        }
        catch (Exception ex)
        {
            QuestionnaireServiceException error = new QuestionnaireServiceException();
            error.title = "Erreur QuestionnaireService : " + errorTxt;
            error.text = error.title + " nomPays " + nomPays;
            error.details = "\n Procédure Stockée : " + procName + " \n " + ex.Message;
            throw new FaultException<QuestionnaireServiceException>(error);
        }
    }


    /// <summary>
    /// Méthode qui va insérer une ville dans la base de données EX: 3960 Sierre
    /// </summary>
    /// <param name="idPays">identifiant du pays où se trouve la ville</param>
    /// <param name="npVille">Numéro Postal de la ville</param>
    /// <param name="nomVille">Nom de la ville</param>
    void QuestionnaireContract.insertVille(int idPays, int npVille, string nomVille)
    {
        procName = "sp_insertVille";
        errorTxt = "Insertion d'une Ville";
        con = new DbConnectionManager();

        try
        {
            //définition de la procédure stockée
            con.useProc(procName);

            //Ajout de paramètres
            con.Com.Parameters.Add("@ID_PAYS", SqlDbType.Int).Value = idPays;
            con.Com.Parameters.Add("@NP_VILLE", SqlDbType.Int).Value = npVille;
            con.Com.Parameters.Add("@NOMVILLE", SqlDbType.VarChar).Value = nomVille;

            //Execution de la procédure
            con.execProc();

        }
        catch (Exception ex)
        {
            QuestionnaireServiceException error = new QuestionnaireServiceException();
            error.title = "Erreur QuestionnaireService : " + errorTxt;
            error.text = error.title + " idPays = " + idPays + " npVille = " + npVille + " nomVille = " +
nomVille;
            error.details = "\n Procédure Stockée : " + procName + " \n " + ex.Message;
            throw new FaultException<QuestionnaireServiceException>(error);
        }
    }

    /// <summary>
    /// Méthode qui va insérer un type de Question Ex: Ouverte
    /// </summary>
    /// <param name="typeQuestion">nom du type de la question</param>
    void QuestionnaireContract.insertTypeQuestion(string typeQuestion)
    {
        procName = "sp_insertTypeQuestion";
        errorTxt = "Insertion d'un type de question";
        con = new DbConnectionManager();
```

```
            try
            {
                //définition de la procédure stockée
                con.useProc(procName);

                //Ajout de paramètres
                con.Com.Parameters.Add("@TYPEQUESTION", SqlDbType.VarChar).Value =
typeQuestion;

                //Execution de la procédure
                con.execProc();

            }
            catch (Exception ex)
            {
                QuestionnaireServiceException error = new QuestionnaireServiceException();
                error.title = "Erreur QuestionnaireService : " + errorTxt;
                error.text = error.title + " typeQuestion = " + typeQuestion ;
                error.details = "\n Procédure Stockée : " + procName + " \n " + ex.Message;
                throw new FaultException<QuestionnaireServiceException>(error);
            }
        }

        /// <summary>
        /// Méthode qui va récupérer une personne en fonction d'un identifiant
        /// </summary>
        /// <param name="idPersonne">identifiant d'une personne</param>
        /// <returns>retourne un dataset contenant la personne récupérée</returns>
        DataSet QuestionnaireContract.getPersonneID(int idPersonne)
        {
            ds = new DataSet();
            procName = "sp_selectPersonneID";
            errorTxt = "Récupération d'un client selon un identifiant";
            con = new DbConnectionManager();

            try
            {
                //définition de la procédure stockée
                con.selectProc(procName);

                //Ajout de paramètres
                con.Com.Parameters.Add("@ID_PERSONNE", SqlDbType.Int).Value = idPersonne;
                ds = con.fillDataset();
            }
            catch (Exception ex)
            {
                QuestionnaireServiceException error = new QuestionnaireServiceException();
                error.title = "Erreur QuestionnaireService : " + errorTxt;
                error.text = error.title + " idPersonne = " + idPersonne;
                error.details = "\n Procédure Stockée : " + procName + " \n " + ex.Message;
                throw new FaultException<QuestionnaireServiceException>(error);
            }

            return ds;
        }

        /// <summary>
        /// Méthode privée utilisée par d'autres méthodes pour insérer une personne dans la base de
données
        /// </summary>
```

```
        private void insertPersonne(int idFonction, int idEtudes, string prenom, string nom, string
sexe, string dateNaissance, string tel, string email, string adresse, int npVille, string entreprise,
string login, string password)
        {
            procName = "sp_insertPersonne";
            errorTxt = "Insertion d'une Personne";
            con = new DbConnectionManager();

            try
            {
                //définition de la procédure stockée
                con.useProc(procName);

                //Ajout de paramètres
                con.Com.Parameters.Add("@NP_VILLE", SqlDbType.Int).Value = npVille;
                con.Com.Parameters.Add("@ID_FONCTION", SqlDbType.Int).Value = idFonction;
                con.Com.Parameters.Add("@ID_ETUDES", SqlDbType.Int).Value = idEtudes;
                con.Com.Parameters.Add("@PRENOM", SqlDbType.VarChar).Value = prenom;
                con.Com.Parameters.Add("@NOM", SqlDbType.VarChar).Value = nom;
                con.Com.Parameters.Add("@SEXE", SqlDbType.VarChar).Value = sexe;
                con.Com.Parameters.Add("@DATENAISSANCE", SqlDbType.VarChar).Value =
dateNaissance;
                con.Com.Parameters.Add("@TEL", SqlDbType.VarChar).Value = tel;
                con.Com.Parameters.Add("@EMAIL", SqlDbType.VarChar).Value = email;
                con.Com.Parameters.Add("@ADRESSE", SqlDbType.VarChar).Value = adresse;
                con.Com.Parameters.Add("@ENTREPRISE", SqlDbType.VarChar).Value = entreprise;
                con.Com.Parameters.Add("@LOGIN", SqlDbType.VarChar).Value = login;
                con.Com.Parameters.Add("@PASSWORD", SqlDbType.VarChar).Value = password;


                //Execution de la procédure
                con.execProc();

            }
            catch (Exception ex)
            {
                QuestionnaireServiceException error = new QuestionnaireServiceException();
                error.title = "Erreur QuestionnaireService : " + errorTxt;
                error.text = error.title + " prenom = " + prenom + " nom = " + nom;
                error.details = "\n Procédure Stockée : " + procName + " \n " + ex.Message;
                throw new FaultException<QuestionnaireServiceException>(error);
            }
        }

        /// /// <summary>
        /// Méthode qui va insérer une personne dans la base de données
        /// </summary>
        /// <param name="idFonction">identifiant de la fonction</param>
        /// <param name="idEtudes">identifiant des études</param>
        /// <param name="prenom">prenom de la personne</param>
        /// <param name="nom">nom de la personne</param>
        /// <param name="sexe">sexe de la personne</param>
        /// <param name="dateNaissance">date de naissance</param>
        /// <param name="tel">numéro de téléphone</param>
        /// <param name="email">adresse email</param>
        /// <param name="adresse">adresse</param>
        /// <param name="npVille">numéro postal</param>
        /// <param name="entreprise">employeur</param>
        /// <param name="login">login</param>
        /// <param name="password">mot de passe</param>
```

```
        void QuestionnaireContract.insertPersonne(int idFonction, int idEtudes, string prenom, string
nom, string sexe, string dateNaissance, string tel, string email, string adresse, int npVille, string
entreprise, string login, string password)
        {
            insertPersonne(idFonction, idEtudes, prenom, nom, sexe, dateNaissance, tel, email,
adresse, npVille, entreprise, login, password);
        }

        /// <summary>
        /// Méthode qui va insérer une personne dans la base de données avec un object Personne
        /// </summary>
        /// <param name="pers">Object Personne</param>
        void QuestionnaireContract.insertPersonneObject(Personne pers)
        {
            insertPersonne(pers.idFonction, pers.idEtudes, pers.prenom, pers.nom, pers.sexe,
pers.dateNaissance, pers.tel, pers.email, pers.adresse, pers.npVille, pers.entreprise, pers.login,
pers.password);
        }


        /// <summary>
        /// Méthode qui va récupérer tous les questionnaires
        /// </summary>
        /// <returns>retourne un Dataset Contenant tous les questionnaires</returns>
        DataSet QuestionnaireContract.getQuestionnaires()
        {
            ds = new DataSet();
            procName = "sp_selectQuestionnaires";
            errorTxt = "Récupération de toues les questionnaires";
            con = new DbConnectionManager();

            try
            {
                ds = con.selectProcWithoutParam(procName);
            }
            catch (Exception ex)
            {
                QuestionnaireServiceException error = new QuestionnaireServiceException();
                error.title = "Erreur QuestionnaireService : " + errorTxt;
                error.text = error.title;
                error.details = "\n Procédure Stockée : " + procName + " \n " + ex.Message;
                throw new FaultException<QuestionnaireServiceException>(error);
            }

            return ds;

        }

        /// <summary>
        /// Méthode privée utilisée par d'autres méthodes pour insérer un questionnaire dans la base
de données
        /// </summary>
        private void insertQuestionnaire(int idAuteur, string titreQuestionnaire, string description,
string objectif)
        {
            procName = "sp_insertQuestionnaire";
            errorTxt = "Insertion d'un Questionnaire";
            con = new DbConnectionManager();

            try
            {
                //définition de la procédure stockée
```

```
            con.useProc(procName);

            //Ajout de paramètres
            con.Com.Parameters.Add("@ID_AUTEUR", SqlDbType.Int).Value = idAuteur;
            con.Com.Parameters.Add("@TITREQUESTIONNAIRE", SqlDbType.VarChar).Value =
titreQuestionnaire;
            con.Com.Parameters.Add("@DESCRIPTION", SqlDbType.Text).Value = description;
            con.Com.Parameters.Add("@OBJECTIF", SqlDbType.Text).Value = objectif;

            //Execution de la procédure
            con.execProc();

        }
        catch (Exception ex)
        {
            QuestionnaireServiceException error = new QuestionnaireServiceException();
            error.title = "Erreur QuestionnaireService : " + errorTxt;
            error.text = error.title + " titre questionnaire = " + titreQuestionnaire;
            error.details = "\n Procédure Stockée : " + procName + " \n " + ex.Message;
            throw new FaultException<QuestionnaireServiceException>(error);
        }
    }

    /// <summary>
    /// Méthode qui insérer un questionnaire dans la base de données EX: Comment menez-
vous vos projets ?
    /// </summary>
    /// <param name="idAuteur">identifiant d'un personne étant l'auteur</param>
    /// <param name="titreQuestionnaire">titre du questionnaire</param>
    /// <param name="description">description du questionnaire</param>
    /// <param name="objectif">objectfis du questionnaire</param>
    void QuestionnaireContract.insertQuestionnaire(int idAuteur, string titreQuestionnaire, string
description, string objectif)
    {
        insertQuestionnaire(idAuteur, titreQuestionnaire,description,objectif);
    }


    /// <summary>
    /// Méthode qui insérer un questionnaire dans la base de données selon un object de type
Questionnare
    /// </summary>
    /// <param name="quest">object Questionnaire</param>
    void QuestionnaireContract.insertQuestionnaireObject(Questionnaire quest)
    {
        insertQuestionnaire(quest.idAuteur, quest.titreQuestionnaire, quest.description,
quest.objectifs);
    }


    /// <summary>
    /// Méthode qui va récupérer les questions d'un questionnaire
    /// </summary>
    /// <param name="idQuestionnaire">identifiant d'un questionnaire</param>
    /// <returns>retourne un dataset contenant toutes les questions d'un questionnaire</returns>
    DataSet QuestionnaireContract.getQuestions(int idQuestionnaire)
    {
        ds = new DataSet();
        procName = "sp_selectQuestions";
        errorTxt = "Récupération des questions selon un questionnaire";
        con = new DbConnectionManager();
```

```
            try
            {
                //définition de la procédure stockée
                con.selectProc(procName);

                //Ajout de paramètres
                con.Com.Parameters.Add("@ID_QUESTIONNAIRE", SqlDbType.Int).Value =
idQuestionnaire;
                ds = con.fillDataset();
            }
            catch (Exception ex)
            {
                QuestionnaireServiceException error = new QuestionnaireServiceException();
                error.title = "Erreur QuestionnaireService : " + errorTxt;
                error.text = error.title + "idQuestionnaire = " + idQuestionnaire;
                error.details = "\n Procédure Stockée : " + procName + " \n " + ex.Message;
                throw new FaultException<QuestionnaireServiceException>(error);
            }

            return ds;
        }

        /// <summary>
        /// Méthode privée utilisée par d'autres méthodes pour insérer une question dans la base de
données
        /// </summary>
        private void insertQuestion(int idQuestionnaire, int idTypeQuestion, string question, string
urlQuestion)
        {
            procName = "sp_insertQuestion";
            errorTxt = "Insertion d'une Question";
            con = new DbConnectionManager();

            try
            {
                //définition de la procédure stockée
                con.useProc(procName);

                //Ajout de paramètres
                con.Com.Parameters.Add("@ID_QUESTIONAIRE", SqlDbType.Int).Value =
idQuestionnaire;
                con.Com.Parameters.Add("@ID_TYPEQUESTION", SqlDbType.Int).Value =
idTypeQuestion;
                con.Com.Parameters.Add("@QUESTION", SqlDbType.VarChar).Value = question;
                con.Com.Parameters.Add("@URLQUESTION", SqlDbType.VarChar).Value =
urlQuestion;

                //Execution de la procédure
                con.execProc();

            }
            catch (Exception ex)
            {
                QuestionnaireServiceException error = new QuestionnaireServiceException();
                error.title = "Erreur QuestionnaireService : " + errorTxt;
                error.text = error.title + " question = " + question;
                error.details = "\n Procédure Stockée : " + procName + " \n " + ex.Message;
                throw new FaultException<QuestionnaireServiceException>(error);
            }
        }

        /// <summary>
```

```
        /// Méthode qui va insérer une question dans la base de données EX: Vous choisissez
principalement vos projets en fonction de :
        /// </summary>
        /// <param name="idQuestionnaire">identifiant de la question</param>
        /// <param name="idTypeQuestion">identifiant du type de question</param>
        /// <param name="question">question à poser</param>
        /// <param name="urlQuestion">url pour afficher une image</param>
        void QuestionnaireContract.insertQuestion(int idQuestionnaire, int idTypeQuestion, string
question, string urlQuestion)
        {
            insertQuestion(idQuestionnaire, idTypeQuestion, question, urlQuestion);
        }

        /// <summary>
        /// Méthode qui va insérer une question dans la base de données selon un object de type
Question
        /// </summary>
        /// <param name="quest">object Question</param>
        void QuestionnaireContract.insertQuestionObject(Question quest)
        {
            insertQuestion(quest.idQuestionnaire, quest.idTypeQuestion, quest.question,
quest.urlQuestion);
        }



        /// <summary>
        /// Méthode privée utilisée par d'autres méthodes pour insérer un questionnaire dans la base
de données
        /// </summary>
        private void insertReponse(int idQuestion, string reponse, int points)
        {
            procName = "sp_insertReponse";
            errorTxt = "Insertion d'une Réponse";
            con = new DbConnectionManager();

            try
            {
                //définition de la procédure stockée
                con.useProc(procName);

                //Ajout de paramètres
                con.Com.Parameters.Add("@IDQUESTION", SqlDbType.Int).Value = idQuestion;
                con.Com.Parameters.Add("@REPONSE", SqlDbType.VarChar).Value = reponse;
                con.Com.Parameters.Add("@POINTS", SqlDbType.Int).Value = points;

                //Execution de la procédure
                con.execProc();

            }
            catch (Exception ex)
            {
                QuestionnaireServiceException error = new QuestionnaireServiceException();
                error.title = "Erreur QuestionnaireService : " + errorTxt;
                error.text = error.title + " identifiant Question = " + idQuestion;
                error.details = "\n Procédure Stockée : " + procName + " \n " + ex.Message;
                throw new FaultException<QuestionnaireServiceException>(error);
            }
        }

        /// <summary>
        /// Méthode qui va récupérer les réponses en fonction d'une question
```

```
/// </summary>
/// <param name="idQuestion">identifiant d'une question</param>
/// <returns>retourne un dataset contenant toutes les réponses</returns>
DataSet QuestionnaireContract.getReponses(int idQuestion)
{
    ds = new DataSet();
    procName = "sp_selectReponses";
    errorTxt = "Récupération des réponses selon une question";
    con = new DbConnectionManager();

    try
    {
        //définition de la procédure stockée
        con.selectProc(procName);

        //Ajout de paramètres
        con.Com.Parameters.Add("@ID_QUESTION", SqlDbType.Int).Value = idQuestion;
        ds = con.fillDataset();
    }
    catch (Exception ex)
    {
        QuestionnaireServiceException error = new QuestionnaireServiceException();
        error.title = "Erreur QuestionnaireService : " + errorTxt;
        error.text = error.title + " identifiant question =" + idQuestion;
        error.details = "\n Procédure Stockée : " + procName + " \n " + ex.Message;
        throw new FaultException<QuestionnaireServiceException>(error);
    }

    return ds;
}

/// <summary>
/// Méthode qui va retourne les poits d'une réponse
/// </summary>
/// <param name="idReponse">identifiant de la réponse</param>
/// <returns>int représentant les points de la réponse</returns>
int QuestionnaireContract.getPointReponse(int idReponse)
{
    int points = -1;
    procName = "sp_selectPointReponse";
    errorTxt = "Récupération des points d'une réponse";
    con = new DbConnectionManager();

    try
    {
        //définition de la procédure stockée
        con.useProc(procName);

        //Ajout de paramètres
        con.Com.Parameters.Add("@ID_REPONSE", SqlDbType.Int).Value = idReponse;

        //Execution de la procédure
        points = Convert.ToInt32(con.execScalar());

    }
    catch (Exception ex)
    {
        QuestionnaireServiceException error = new QuestionnaireServiceException();
        error.title = "Erreur QuestionnaireService : " + errorTxt;
        error.text = error.title + " identifiant reponse = " + idReponse;
        error.details = "\n Procédure Stockée : " + procName + " \n " + ex.Message;
        throw new FaultException<QuestionnaireServiceException>(error);
```

```
            }
            return points;
        }

        /// <summary>
        /// Méthode qui va insérer une réponse dans la base de données EX:leur attraction
        /// </summary>
        /// <param name="idQuestion">identifiant de la question</param>
        /// <param name="reponse">reponse à proposer</param>
        /// <param name="points">points de la réponse ou notation appropriée au module
métier</param>
        void QuestionnaireContract.insertReponse(int idQuestion, string reponse, int points)
        {
            insertReponse(idQuestion, reponse, points);
        }

        /// <summary>
        /// Méthode qui va insérer une réponse dans la base de données selon un object Reponse
        /// </summary>
        /// <param name="r">Reponse r</param>
        void QuestionnaireContract.insertReponseObject(Reponse r)
        {
            insertReponse(r.idQuestion, r.reponse, r.points);
        }

        /// <summary>
        /// Méthode privée utilisée par d'autres méthodes pour insérer une réponse ouverte dans la
base de données
        /// </summary>
        private void insertReponseOuverte(int idReponse, string ReponseOuverte)
        {
            procName = "sp_insertReponseOuverte";
            errorTxt = "Insertion d'une Réponse Ouverte";
            con = new DbConnectionManager();

            try
            {
                //définition de la procédure stockée
                con.useProc(procName);

                //Ajout de paramètres
                con.Com.Parameters.Add("@ID_REPONSE", SqlDbType.Int).Value = idReponse;
                con.Com.Parameters.Add("@REPONSEOUVERTE", SqlDbType.VarChar).Value =
ReponseOuverte;

                //Execution de la procédure
                con.execProc();

            }
            catch (Exception ex)
            {
                QuestionnaireServiceException error = new QuestionnaireServiceException();
                error.title = "Erreur QuestionnaireService : " + errorTxt;
                error.text = error.title + " identifiant reponse = " + idReponse + " Reponse ouverte = " +
ReponseOuverte;
                error.details = "\n Procédure Stockée : " + procName + " \n " + ex.Message;
                throw new FaultException<QuestionnaireServiceException>(error);
            }
        }


        /// <summary>
```

```
        /// Méthode qui va insérer une réponse ouverte dans la base de données EX: je choisi en
général mes projets selon plusieurs critères...
        /// </summary>
        /// <param name="idReponse"></param>
        /// <param name="ReponseOuverte"></param>
        void QuestionnaireContract.insertReponseOuverte(int idReponse, string ReponseOuverte)
        {
            insertReponseOuverte(idReponse, ReponseOuverte);
        }

        /// <summary>
        /// Méthode qui va insérer une réponse ouverte dans la base de données selon un object
ReponseOuverte
        /// </summary>
        /// <param name="r">object ReponseOuverte</param>
        void QuestionnaireContract.insertReponseOuverteObject(ReponseOuverte r)
        {
            insertReponseOuverte(r.idReponse, r.reponseOuverte);
        }

        /// <summary>
        /// Méthode qui va insérer une réponse à une fiche de résultat
        /// Cette action intervien lorsqu'une personne choisi une réponse à un questionnaire
        /// </summary>
        /// <param name="idFicheResultats">indentifiant d'une fiche de résultats</param>
        /// <param name="idReponse">identifiant d'une réponse</param>
        void QuestionnaireContract.insertReponseQuestionnaire(int idFicheResultats, int
idReponse)
        {
            procName = "sp_insertReponseQuestionnaire";
            errorTxt = "Insertion d'une Réponse à un questionnaire";
            con = new DbConnectionManager();

            try
            {
                //définition de la procédure stockée
                con.useProc(procName);

                //Ajout de paramètres
                con.Com.Parameters.Add("@ID_FICHERESULTATS", SqlDbType.Int).Value =
idFicheResultats;
                con.Com.Parameters.Add("@ID_REPONSE", SqlDbType.Int).Value = idReponse;

                //Execution de la procédure
                con.execProc();

            }
            catch (Exception ex)
            {
                QuestionnaireServiceException error = new QuestionnaireServiceException();
                error.title = "Erreur QuestionnaireService : " + errorTxt;
                error.text = error.title + " identifiant ficheResultats = " + idFicheResultats + " identifiant
réponse = " + idReponse ;
                error.details = "\n Procédure Stockée : " + procName + " \n " + ex.Message;
                throw new FaultException<QuestionnaireServiceException>(error);
            }
        }
    }
}
```

Chaque méthode dans cette classe, va faire appel à une procédure stockées de la base de données QuestionnaireDB.

IL est également à noter, que lorsqu'une méthode rencontre un problème, une exception de type FaultException<QuestionnaireServiceException> est retournée.

Vous remarquerez également que pour les interactions avec la base de données, la classe«DbConnectionManager » est utilisée.

Routines sur la base de données

Afin de faciliter la connexion et l'exécution des procédures stockées, la classe « DbConnectionManage » a été créée.

Voici le code de cette classe :

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Configuration ;
using System.Data;
using System.Data.SqlClient;

namespace QuestionnaireService
{
   public class DbConnectionManager
   {

      /// <summary>
      /// Variables
      /// </summary>
      ConnectionStringSettings cfg;
      SqlConnection con ;
      SqlCommand com ;
      SqlDataAdapter sA;
      string cs ;

      /// <summary>
      /// GETTERS SETTERS
      /// </summary>
      public SqlCommand Com
      {
         get { return com; }
         set { com = value; }
      }

      public SqlDataAdapter SA
      {
         get { return sA; }
         set { sA = value ;}
      }

      /// <summary>
      /// Méthode qui va instancier une nouvelle connection
      /// </summary>
      public DbConnectionManager()
      {
         #region SQL Connection
         cfg = ConfigurationManager.ConnectionStrings["QuestionnaireConnectionString"];
         cs = cfg.ConnectionString;
         con = new SqlConnection(cs);
         #endregion
      }

      /// <summary>
      /// Méthode qui va crée une SqlCommand pour exécuter une procédure stockée de type
      /// INSERT UPDATE OU DELETE
      /// </summary>
      /// <param name="procName">nom d'une procédure stockées</param>
```

```
public void useProc(string procName)
{
com = new SqlCommand(procName, con);
com.CommandType = CommandType.StoredProcedure;
}

/// <summary>
/// Méthode qui va exécuter une procédure stockée de type
/// INSERT UPDATE OU DELETE
/// </summary>
/// <param name="procName">nom d'une procédure stockées</param>
public void execProc()
{
    this.open();
    com.ExecuteNonQuery();
    this.close();
}

/// <summary>
/// Méthode qui va exécuter une procédure stokée et renvoyer la première cellule du dataset
/// </summary>
/// <returns>la première ligne du dataset à savoir un object</returns>
public Object execScalar()
{
    this.open();
    Object O = com.ExecuteScalar();
    this.close();
    return O ;
}

/// <summary>
/// Méthode qui va crée une SqlCommand pour exécuter une procédure stockée de type
/// INSERT UPDATE OU DELETE sans paramètres
/// </summary>
/// <param name="procName">nom d'une procédure stockées</param>
public void useProcWithoutParam(string procName)
{
    useProc(procName);
    execProc();
}

/// <summary>
/// Méthode qui va crée un  SqlDataAdapter pour exécuter une procédure stockée de type
/// SELECT
/// </summary>
public void selectProc(string proceName)
{
    useProc(proceName);
    sA = new SqlDataAdapter() ;
    sA.SelectCommand = com ;
}

/// <summary>
/// Méthode qui va remplir un dataset selon un SqlDataAdapter
/// </summary>
/// <returns>Dataset contenant les données récupérées</returns>
public DataSet fillDataset()
{
    DataSet ds = new DataSet();
    this.open();
    this.sA.Fill(ds);
    this.close() ;
```

```
            return ds ;
        }

        /// <summary>
        /// Méthode qui va crée un  SqlDataAdapter pour exécuter une procédure stockée de type
        /// SELECT sans paramètres
        /// </summary>
        public DataSet selectProcWithoutParam(string proceName)
        {
            selectProc(proceName);
            return fillDataset();
        }


        /// <summary>
        /// Méthode qui va ouvrir la connexion à la Base de données
        /// </summary>
        public void open()
        {
            con.Open();
        }

        /// <summary>
        /// Méthode qui va fermer la connexion à la Base de données
        /// </summary>
        public void close()
        {
            con.Close();
        }
    }
}
```

Ficher de configuration

Le fichier de configuration va définir la méthode de communication du web service.

Voici le code de ce fichier :

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
        <connectionStrings>
                <add name="QuestionnaireConnectionString" connectionString="Data
Source=CORE2DUOT5600;Initial Catalog=QuestionnaireDB;Persist Security Info=True;User
ID=adminQuestionnaireDB;Password=admin" providerName="System.Data.SqlClient"/>
        </connectionStrings>
   <system.serviceModel>
      <bindings />
      <behaviors>
         <serviceBehaviors>
            <behavior name="QuestionnaireServiceBehavior">
               <serviceMetadata httpGetEnabled="true" />
               <serviceThrottling maxConcurrentCalls="100" maxConcurrentSessions="100" />
            </behavior>
         </serviceBehaviors>
      </behaviors>
      <services>
         <service behaviorConfiguration="QuestionnaireServiceBehavior"
            name="QuestionnaireService.QuestionnaireContractType">
            <endpoint address="" binding="wsHttpBinding" bindingConfiguration=""
               contract="QuestionnaireService.QuestionnaireContract" />
         </service>
      </services>
   </system.serviceModel>
   <system.web>
      <identity impersonate="false" />
   </system.web>
</configuration>
```

Le section <connectionString> contient les paramètres de connexion à la base de données. Il est important de modifier le nom du serveur SQL par celui que vous utlisez pour les tests.

Il se trouve dans la section Data Source=NOMSERVERSQL.

La section <System.ServiceModel> contient les paramètres propres à la communication WCF. Nous pouvons voir que le binding utilisé est wsHttpBinding. Vous remarquez que l'endpoint n'a pas d'adresse, du au fait que l'adresse est celle du serveur IIS.

En plus de ces trois fichiers, le projet du web service contient également un fichier nommé Questionnaire.svc. Ce fichier permet à IIS de savoir quelle classe doit être exposée en tant que Web Service.

5.5. Client générique avec interface de communication

Vous trouverez le client générique sur le CD de ressources WCF.

Répertoire:
ProjetQuestionnaire\ProjetGénérique\QuestionnaireClientSolution

Ce client est composé de deux projets :

- QuestionnaireClientTools

  Outils génériques pour le client
- QuestionnaireErrorMessageBox

  Formulaire pour l'affichage des erreurs

Afin de faciliter la communication avec le web service du côté client, j'ai créé une interface de communication générique.

Cette interface existe sous forme d'une classe que j'ai nommée « WebServiceManger » et qui se trouve dans le projet « QuestionnaireClientTools ».

Voici le code de cette classe :

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data;
using System.ServiceModel;
using QuestionnaireErrorMessageBox;
using QuestionnaireService ;

namespace QuestionnaireClientTools
{
   public  class WebServiceManager
   {
      /// <summary>
      /// variables
      /// </summary>
      QuestionnaireContractClient proxy;
      DataSet ds;


      /// <summary>
      /// Méhtode qui va récupérer toutes les personnes de la base de données
      /// </summary>
      /// <returns></returns>
      /// <returns>retourne un dataset contenant toutes les personnes</returns>
      public DataSet getAllPersonnes()
      {
         proxy = new QuestionnaireContractClient();
         ds = new DataSet();

         try
```

```
            {
                ds = proxy.getAllPersonnes();
            }
            catch (FaultException<QuestionnaireServiceException> error)
            {
                ErrorMessageBox.Show(error.Detail.text, error.Detail.details, error.Detail.title);
            }
            catch (EndpointNotFoundException ex)
            {
                ErrorMessageBox.Show("L'application n'a pu trouver le webservice. Vérifiez la
configuration du endpoint dans le fichier App.config.", ex.Message, "Endpoint non trouvé");
            }
            catch (Exception ex)
            {
                ErrorMessageBox.Show("exception rencontrée", ex.Message, "erreur dans le web
service");
            }

            return ds;
        }

        /// <summary>
        /// Méthode qui va récupér une personne en fonction de son login et mot de passe
        /// </summary>
        /// <param name="login">login de la personne</param>
        /// <param name="psw">mot de passe de la personne</param>
        /// <returns>retourne un dataset contenant la personne récupérée</returns>
        public DataSet getPersonne(string login, string psw)
        {
            proxy = new QuestionnaireContractClient();
            ds = new DataSet();

            try
            {
                ds = proxy.getPersonne(login, psw);
            }
            catch (FaultException<QuestionnaireServiceException> error)
            {
                ErrorMessageBox.Show(error.Detail.text, error.Detail.details, error.Detail.title);
            }
            catch (EndpointNotFoundException ex)
            {
                ErrorMessageBox.Show("L'application n'a pu trouver le webservice. Vérifiez la
configuration du endpoint dans le fichier App.config.", ex.Message, "Endpoint non trouvé");
            }
            catch (Exception ex)
            {
                ErrorMessageBox.Show("exception rencontrée", ex.Message, "erreur dans le web
service");
            }

            return ds;
        }


        /// <summary>
        /// Méthode qui va récupérer toutes le fiches de résultat d'une personne
        /// </summary>
        /// <param name="idPersonne">identifiant de la personne</param>
        /// <returns>retourne un dataset avec les fiches de résultats</returns>
        public DataSet getFichesResultatsPersonne(int idPersonne)
        {
```

```
            proxy = new QuestionnaireContractClient();
            ds = new DataSet();

            try
            {
                ds = proxy.getFichesResultatsPersonne(idPersonne);
            }
            catch (FaultException<QuestionnaireServiceException> error)
            {
                ErrorMessageBox.Show(error.Detail.text, error.Detail.details, error.Detail.title);
            }
            catch (EndpointNotFoundException ex)
            {
                ErrorMessageBox.Show("L'application n'a pu trouver le webservice. Vérifiez la
configuration du endpoint dans le fichier App.config.", ex.Message, "Endpoint non trouvé");
            }
            catch (Exception ex)
            {
                ErrorMessageBox.Show("exception rencontrée", ex.Message, "erreur dans le web
service");
            }

            return ds;
        }

        /// <summary>
        /// Méthode qui va récupérer une fiche de résultats
        /// </summary>
        /// <param name="idFicheResultat">identifiant de la fiche de résultars</param>
        /// <returns>dataset contenant la fiche de résultat de la personne</returns>
        public DataSet getFicheResultats(int idFicheResultat)
        {
            proxy = new QuestionnaireContractClient();
            ds = new DataSet();

            try
            {
                ds = proxy.getFicheResultats(idFicheResultat);
            }
            catch (FaultException<QuestionnaireServiceException> error)
            {
                ErrorMessageBox.Show(error.Detail.text, error.Detail.details, error.Detail.title);
            }
            catch (EndpointNotFoundException ex)
            {
                ErrorMessageBox.Show("L'application n'a pu trouver le webservice. Vérifiez la
configuration du endpoint dans le fichier App.config.", ex.Message, "Endpoint non trouvé");
            }
            catch (Exception ex)
            {
                ErrorMessageBox.Show("exception rencontrée", ex.Message, "erreur dans le web
service");
            }

            return ds;
        }

        /// <summary>
        /// Méthode qui va insérer le résultat d'un questionnaire à une fiche de résultat
        /// </summary>
        /// <param name="idFicheResultat">identifiant de la fiche de résultas</param>
        /// <param name="resultat">résultat</param>
```

```
        public void insertResultat(int idFicheResultat, string resultat)
        {
            proxy = new QuestionnaireContractClient();
            try
            {
                proxy.insertResultat(idFicheResultat, resultat);
            }
            catch (FaultException<QuestionnaireServiceException> error)
            {
                ErrorMessageBox.Show(error.Detail.text, error.Detail.details, error.Detail.title);
            }
            catch (EndpointNotFoundException ex)
            {
                ErrorMessageBox.Show("L'application n'a pu trouver le webservice. Vérifiez la
configuration du endpoint dans le fichier App.config.", ex.Message, "Endpoint non trouvé");
            }
            catch (Exception ex)
            {
                ErrorMessageBox.Show("exception rencontrée", ex.Message, "erreur dans le web
service");
            }
        }


        /// <summary>
        /// Méthode qui insérer un type d'étude dans la base de données EX: HES
        /// </summary>
        /// <param name="etudes">nom de l'étude</param>
        public void insertEtudes(string etudes)
        {
            proxy = new QuestionnaireContractClient();
            try
            {
                proxy.insertEtudes(etudes);
            }
            catch (FaultException<QuestionnaireServiceException> error)
            {
                ErrorMessageBox.Show(error.Detail.text, error.Detail.details, error.Detail.title);
            }
            catch (EndpointNotFoundException ex)
            {
                ErrorMessageBox.Show("L'application n'a pu trouver le webservice. Vérifiez la
configuration du endpoint dans le fichier App.config.", ex.Message, "Endpoint non trouvé");
            }
            catch (Exception ex)
            {
                ErrorMessageBox.Show("exception rencontrée", ex.Message, "erreur dans le web
service");
            }
        }


        /// <summary>
        /// Méthode qui va crée une fiche de résultat pour une personne
        /// </summary>
        /// <param name="idPersonne">identifiant de la personne</param>
        public int insertFicheResultats(int idPersonne)
        {
            int idFiche = -1;
            proxy = new QuestionnaireContractClient();
            try
            {
```

```
            idFiche = proxy.insertFicheResultats(idPersonne);
        }
        catch (FaultException<QuestionnaireServiceException> error)
        {
            ErrorMessageBox.Show(error.Detail.text, error.Detail.details, error.Detail.title);
        }
        catch (EndpointNotFoundException ex)
        {
            ErrorMessageBox.Show("L'application n'a pu trouver le webservice. Vérifiez la
configuration du endpoint dans le fichier App.config.", ex.Message, "Endpoint non trouvé");
        }
        catch (Exception ex)
        {
            ErrorMessageBox.Show("exception rencontrée", ex.Message, "erreur dans le web
service");
        }

        return idFiche;
    }


    /// <summary>
    /// Méthode qui va insérer une fonction dans la base de données EX: Informaticien
    /// </summary>
    /// <param name="nomFonction">nom de la fonction</param>
    public void insertFonction(string nomFonction)
    {
        proxy = new QuestionnaireContractClient();
        try
        {
            proxy.insertFonction(nomFonction);
        }
        catch (FaultException<QuestionnaireServiceException> error)
        {
            ErrorMessageBox.Show(error.Detail.text, error.Detail.details, error.Detail.title);
        }
        catch (EndpointNotFoundException ex)
        {
            ErrorMessageBox.Show("L'application n'a pu trouver le webservice. Vérifiez la
configuration du endpoint dans le fichier App.config.", ex.Message, "Endpoint non trouvé");
        }
        catch (Exception ex)
        {
            ErrorMessageBox.Show("exception rencontrée", ex.Message, "erreur dans le web
service");
        }
    }


    /// <summary>
    /// Méthode qui va insérer un pays dans la base de données EX: Suisse
    /// </summary>
    /// <param name="nomPays">nom du pays</param>
    public void insertPays(string nomPays)
    {
        proxy = new QuestionnaireContractClient();
        try
        {
            proxy.insertPays(nomPays);
        }
        catch (FaultException<QuestionnaireServiceException> error)
        {
```

```
            ErrorMessageBox.Show(error.Detail.text, error.Detail.details, error.Detail.title);
        }
        catch (EndpointNotFoundException ex)
        {
            ErrorMessageBox.Show("L'application n'a pu trouver le webservice. Vérifiez la
configuration du endpoint dans le fichier App.config.", ex.Message, "Endpoint non trouvé");
        }
        catch (Exception ex)
        {
            ErrorMessageBox.Show("exception rencontrée", ex.Message, "erreur dans le web
service");
        }
    }


    /// <summary>
    /// Méthode qui va insérer une ville dans la base de données EX: 3960 Sierre
    /// </summary>
    /// <param name="idPays">identifiant du pays où se trouve la ville</param>
    /// <param name="npVille">Numéro Postal de la ville</param>
    /// <param name="nomVille">Nom de la ville</param>
    public void insertVille(int idPays, int npVille, string nomVille)
    {
        proxy = new QuestionnaireContractClient();
        try
        {
            proxy.insertVille(idPays, npVille, nomVille);
        }
        catch (FaultException<QuestionnaireServiceException> error)
        {
            ErrorMessageBox.Show(error.Detail.text, error.Detail.details, error.Detail.title);
        }
        catch (EndpointNotFoundException ex)
        {
            ErrorMessageBox.Show("L'application n'a pu trouver le webservice. Vérifiez la
configuration du endpoint dans le fichier App.config.", ex.Message, "Endpoint non trouvé");
        }
        catch (Exception ex)
        {
            ErrorMessageBox.Show("exception rencontrée", ex.Message, "erreur dans le web
service");
        }
    }


    /// <summary>
    /// Méthode qui va insérer un type de Question Ex: Ouverte
    /// </summary>
    /// <param name="typeQuestion">nom du type de la question</param>
    public void insertTypeQuestion(string typeQuestion)
    {
        proxy = new QuestionnaireContractClient();
        try
        {
            proxy.insertTypeQuestion(typeQuestion);
        }
        catch (FaultException<QuestionnaireServiceException> error)
        {
            ErrorMessageBox.Show(error.Detail.text, error.Detail.details, error.Detail.title);
        }
        catch (EndpointNotFoundException ex)
        {
```

```
            ErrorMessageBox.Show("L'application n'a pu trouver le webservice. Vérifiez la
configuration du endpoint dans le fichier App.config.", ex.Message, "Endpoint non trouvé");
        }
        catch (Exception ex)
        {
            ErrorMessageBox.Show("exception rencontrée", ex.Message, "erreur dans le web
service");
        }
    }


    /// <summary>
    /// Méthode qui va récupérer une personne en fonction d'un identifiant
    /// </summary>
    /// <param name="idPersonne">identifiant d'une personne</param>
    /// <returns>retourne un dataset contenant la personne récupérée</returns>
    public DataSet getPersonneID(int idPersonne)
    {
        proxy = new QuestionnaireContractClient();
        ds = new DataSet();

        try
        {
            ds = proxy.getPersonneID(idPersonne);
        }
        catch (FaultException<QuestionnaireServiceException> error)
        {
            ErrorMessageBox.Show(error.Detail.text, error.Detail.details, error.Detail.title);
        }
        catch (EndpointNotFoundException ex)
        {
            ErrorMessageBox.Show("L'application n'a pu trouver le webservice. Vérifiez la
configuration du endpoint dans le fichier App.config.", ex.Message, "Endpoint non trouvé");
        }
        catch (Exception ex)
        {
            ErrorMessageBox.Show("exception rencontrée", ex.Message, "erreur dans le web
service");
        }

        return ds;
    }

    /// /// <summary>
    /// Méthode qui va insérer une personne dans la base de donées
    /// </summary>
    /// <param name="idFonction">identifiant de la fonction</param>
    /// <param name="idEtudes">identifiant des études</param>
    /// <param name="prenom">prenom de la personne</param>
    /// <param name="nom">nom de la personne</param>
    /// <param name="sexe">sexe de la personne</param>
    /// <param name="dateNaissance">date de naissance</param>
    /// <param name="tel">numéro de téléphone</param>
    /// <param name="email">adresse email</param>
    /// <param name="adresse">adresse</param>
    /// <param name="npVille">numéro postal</param>
    /// <param name="entreprise">employeur</param>
    /// <param name="login">login</param>
    /// <param name="password">mot de passe</param>
    public void insertPersonne(int idFonction, int idEtudes, string prenom, string nom, string
sexe, string dateNaissance, string tel, string email, string adresse, int npVille, string entreprise,
string login, string password)
```

```
        {
            proxy = new QuestionnaireContractClient();
            try
            {
                proxy.insertPersonne(idFonction, idEtudes, prenom, nom, sexe, dateNaissance, tel,
email, adresse, npVille, entreprise, login, password);
            }
            catch (FaultException<QuestionnaireServiceException> error)
            {
                ErrorMessageBox.Show(error.Detail.text, error.Detail.details, error.Detail.title);
            }
            catch (EndpointNotFoundException ex)
            {
                ErrorMessageBox.Show("L'application n'a pu trouver le webservice. Vérifiez la
configuration du endpoint dans le fichier App.config.", ex.Message, "Endpoint non trouvé");
            }
            catch (Exception ex)
            {
                ErrorMessageBox.Show("exception rencontrée", ex.Message, "erreur dans le web
service");
            }
        }


        /// <summary>
        /// Méthode qui va insérer une personne dans la base de données avec un object Personne
        /// </summary>
        /// <param name="pers">Object Personne</param>
        public void insertPersonneObject(Personne pers)
        {
            proxy = new QuestionnaireContractClient();
            try
            {
                proxy.insertPersonneObject(pers);
            }
            catch (FaultException<QuestionnaireServiceException> error)
            {
                ErrorMessageBox.Show(error.Detail.text, error.Detail.details, error.Detail.title);
            }
            catch (EndpointNotFoundException ex)
            {
                ErrorMessageBox.Show("L'application n'a pu trouver le webservice. Vérifiez la
configuration du endpoint dans le fichier App.config.", ex.Message, "Endpoint non trouvé");
            }
            catch (Exception ex)
            {
                ErrorMessageBox.Show("exception rencontrée", ex.Message, "erreur dans le web
service");
            }
        }


        /// <summary>
        /// Méthode qui va récupérer tous les questionnaires
        /// </summary>
        /// <returns>retourne un Dataset Contenant tous les questionnaires</returns>
        public DataSet getQuestionnaires()
        {
            proxy = new QuestionnaireContractClient();
            ds = new DataSet();

            try
```

```
            {
                ds = proxy.getQuestionnaires();
            }
            catch (FaultException<QuestionnaireServiceException> error)
            {
                ErrorMessageBox.Show(error.Detail.text, error.Detail.details, error.Detail.title);
            }
            catch (EndpointNotFoundException ex)
            {
                ErrorMessageBox.Show("L'application n'a pu trouver le webservice. Vérifiez la
configuration du endpoint dans le fichier App.config.", ex.Message, "Endpoint non trouvé");
            }
            catch (Exception ex)
            {
                ErrorMessageBox.Show("exception rencontrée", ex.Message, "erreur dans le web
service");
            }

            return ds;
        }

        /// <summary>
        /// Méthode qui insérer un questionnaire dans la base de données EX: Comment menez-
vous vos projets ?
        /// </summary>
        /// <param name="idAuteur">identifiant d'un personne étant l'auteur</param>
        /// <param name="titreQuestionnaire">titre du questionnaire</param>
        ///<param name="description">description du questionnaire</param>
        /// <param name="objectif">objectif du questionnaire</param>
        public void insertQuestionnaire(int idAuteur, string titreQuestionnaire, string description,
string objectif)
        {
            proxy = new QuestionnaireContractClient();
            try
            {
                proxy.insertQuestionnaire(idAuteur, titreQuestionnaire,description,objectif);
            }
            catch (FaultException<QuestionnaireServiceException> error)
            {
                ErrorMessageBox.Show(error.Detail.text, error.Detail.details, error.Detail.title);
            }
            catch (EndpointNotFoundException ex)
            {
                ErrorMessageBox.Show("L'application n'a pu trouver le webservice. Vérifiez la
configuration du endpoint dans le fichier App.config.", ex.Message, "Endpoint non trouvé");
            }
            catch (Exception ex)
            {
                ErrorMessageBox.Show("exception rencontrée", ex.Message, "erreur dans le web
service");
            }
        }

        /// <summary>
        /// Méthode qui insére un questionnaire dans la base de données selon un object de type
Questionnare
        /// </summary>
        /// <param name="quest">object Questionnaire</param>
        public void insertQuestionnaireObject(Questionnaire quest)
        {
            proxy = new QuestionnaireContractClient();
            try
```

```
            {
                proxy.insertQuestionnaireObject(quest);
            }
            catch (FaultException<QuestionnaireServiceException> error)
            {
                ErrorMessageBox.Show(error.Detail.text, error.Detail.details, error.Detail.title);
            }
            catch (EndpointNotFoundException ex)
            {
                ErrorMessageBox.Show("L'application n'a pu trouver le webservice. Vérifiez la
configuration du endpoint dans le fichier App.config.", ex.Message, "Endpoint non trouvé");
            }
            catch (Exception ex)
            {
                ErrorMessageBox.Show("exception rencontrée", ex.Message, "erreur dans le web
service");
            }
        }

         /// <summary>
        /// Méthode qui va récupérer les questions d'un questionnaire
        /// </summary>
        /// <param name="idQuestionnaire">identifiant d'un questionnaire</param>
        /// <returns>retourne un dataset contenant toutes les questions d'un questionnaire</returns>
        public DataSet getQuestions(int idQuestionnaire)
        {
            proxy = new QuestionnaireContractClient();
            ds = new DataSet();

            try
            {
                ds = proxy.getQuestions(idQuestionnaire);
            }
            catch (FaultException<QuestionnaireServiceException> error)
            {
                ErrorMessageBox.Show(error.Detail.text, error.Detail.details, error.Detail.title);
            }
            catch (EndpointNotFoundException ex)
            {
                ErrorMessageBox.Show("L'application n'a pu trouver le webservice. Vérifiez la
configuration du endpoint dans le fichier App.config.", ex.Message, "Endpoint non trouvé");
            }
            catch (Exception ex)
            {
                ErrorMessageBox.Show("exception rencontrée", ex.Message, "erreur dans le web
service");
            }

            return ds;
        }

        // <summary>
        /// Méthode qui va insérer une question dans la base de données EX: Vous choisissez
principalement vos projets en fonction de :
        /// </summary>
        /// <param name="idQuestionnaire">identifiant de la question</param>
        /// <param name="idTypeQuestion">identifiant du type de question</param>
        /// <param name="question">question à poser</param>
        /// <param name="urlQuestion">url pour afficher une image</param>
        public void insertQuestion(int idQuestionnaire, int idTypeQuestion, string question, string
urlQuestion)
        {
```

```
            proxy = new QuestionnaireContractClient();
            try
            {
                proxy.insertQuestion(idQuestionnaire, idTypeQuestion, question, urlQuestion);
            }
            catch (FaultException<QuestionnaireServiceException> error)
            {
                ErrorMessageBox.Show(error.Detail.text, error.Detail.details, error.Detail.title);
            }
            catch (EndpointNotFoundException ex)
            {
                ErrorMessageBox.Show("L'application n'a pu trouver le webservice. Vérifiez la
configuration du endpoint dans le fichier App.config.", ex.Message, "Endpoint non trouvé");
            }
            catch (Exception ex)
            {
                ErrorMessageBox.Show("exception rencontrée", ex.Message, "erreur dans le web
service");
            }
        }


        /// <summary>
        /// Méthode qui va insérer une question dans la base de données selon un object de type
Question
        /// </summary>
        /// <param name="quest">object Question</param>
        public void insertQuestionObject(Question quest)
        {
            proxy = new QuestionnaireContractClient();
            try
            {
                proxy.insertQuestionObject(quest);
            }
            catch (FaultException<QuestionnaireServiceException> error)
            {
                ErrorMessageBox.Show(error.Detail.text, error.Detail.details, error.Detail.title);
            }
            catch (EndpointNotFoundException ex)
            {
                ErrorMessageBox.Show("L'application n'a pu trouver le webservice. Vérifiez la
configuration du endpoint dans le fichier App.config.", ex.Message, "Endpoint non trouvé");
            }
            catch (Exception ex)
            {
                ErrorMessageBox.Show("exception rencontrée", ex.Message, "erreur dans le web
service");
            }
        }

        /// <summary>
        /// Méthode qui va récupérer les réponses en fonction d'une question
        /// </summary>
        /// <param name="idQuestion">identifiant d'une question</param>
        /// <returns>retourne un dataset contenant toutes les réponses</returns>
        public DataSet getReponses(int idQuestion)
        {
            proxy = new QuestionnaireContractClient();
            ds = new DataSet();

            try
            {
```

```
            ds = proxy.getReponses(idQuestion);
        }
        catch (FaultException<QuestionnaireServiceException> error)
        {
            ErrorMessageBox.Show(error.Detail.text, error.Detail.details, error.Detail.title);
        }
        catch (EndpointNotFoundException ex)
        {
            ErrorMessageBox.Show("L'application n'a pu trouver le webservice. Vérifiez la
configuration du endpoint dans le fichier App.config.", ex.Message, "Endpoint non trouvé");
        }
        catch (Exception ex)
        {
            ErrorMessageBox.Show("exception rencontrée", ex.Message, "erreur dans le web
service");
        }

        return ds;
    }

    /// <summary>
    /// Méthode qui va retourne les poits d'une réponse
    /// </summary>
    /// <param name="idReponse">identifiant de la réponse</param>
    /// <returns>int représentant les points de la réponse</returns>
    public int getPointReponse(int idReponse)
    {
        proxy = new QuestionnaireContractClient();
        int points = -1;

        try
        {
            points = proxy.getPointReponse(idReponse);
        }
        catch (FaultException<QuestionnaireServiceException> error)
        {
            ErrorMessageBox.Show(error.Detail.text, error.Detail.details, error.Detail.title);
        }
        catch (EndpointNotFoundException ex)
        {
            ErrorMessageBox.Show("L'application n'a pu trouver le webservice. Vérifiez la
configuration du endpoint dans le fichier App.config.", ex.Message, "Endpoint non trouvé");
        }
        catch (Exception ex)
        {
            ErrorMessageBox.Show("exception rencontrée", ex.Message, "erreur dans le web
service");
        }

        return points;
    }

    /// <summary>
    /// Méthode qui va insérer une réponse dans la base de données EX:leur attraction
    /// </summary>
    /// <param name="idQuestion">identifiant de la question</param>
    /// <param name="reponse">reponse à proposer</param>
    /// <param name="points">points de la réponse ou notation appropriée au module
métier</param>
    public void insertReponse(int idQuestion, string reponse, int points)
    {
        proxy = new QuestionnaireContractClient();
```

```
            try
            {
                proxy.insertReponse(idQuestion, reponse, points);
            }
            catch (FaultException<QuestionnaireServiceException> error)
            {
                ErrorMessageBox.Show(error.Detail.text, error.Detail.details, error.Detail.title);
            }
            catch (EndpointNotFoundException ex)
            {
                ErrorMessageBox.Show("L'application n'a pu trouver le webservice. Vérifiez la
configuration du endpoint dans le fichier App.config.", ex.Message, "Endpoint non trouvé");
            }
            catch (Exception ex)
            {
                ErrorMessageBox.Show("exception rencontrée", ex.Message, "erreur dans le web
service");
            }
        }


        /// <summary>
        /// Méthode qui va insérer une réponse dans la base de données selon un object Reponse
        /// </summary>
        /// <param name="r">Reponse r</param>
        public void insertReponseObject(Reponse r)
        {
        proxy = new QuestionnaireContractClient();
            try
            {
                proxy.insertReponseObject(r);
            }
            catch (FaultException<QuestionnaireServiceException> error)
            {
                ErrorMessageBox.Show(error.Detail.text, error.Detail.details, error.Detail.title);
            }
            catch (EndpointNotFoundException ex)
            {
                ErrorMessageBox.Show("L'application n'a pu trouver le webservice. Vérifiez la
configuration du endpoint dans le fichier App.config.", ex.Message, "Endpoint non trouvé");
            }
            catch (Exception ex)
            {
                ErrorMessageBox.Show("exception rencontrée", ex.Message, "erreur dans le web
service");
            }
        }


        /// <summary>
        /// Méthode qui va insérer une réponse ouverte dans la base de données EX: je choisi en
général mes projets selon plusieurs critères...
        /// </summary>
        /// <param name="idReponse"></param>
        /// <param name="ReponseOuverte"></param>
        public void insertReponseOuverte(int idReponse, string ReponseOuverte)
        {
        proxy = new QuestionnaireContractClient();
            try
            {
                proxy.insertReponseOuverte(idReponse, ReponseOuverte);
            }
```

```
            catch (FaultException<QuestionnaireServiceException> error)
            {
                ErrorMessageBox.Show(error.Detail.text, error.Detail.details, error.Detail.title);
            }
            catch (EndpointNotFoundException ex)
            {
                ErrorMessageBox.Show("L'application n'a pu trouver le webservice. Vérifiez la
configuration du endpoint dans le fichier App.config.", ex.Message, "Endpoint non trouvé");
            }
            catch (Exception ex)
            {
                ErrorMessageBox.Show("exception rencontrée", ex.Message, "erreur dans le web
service");
            }
        }


        /// <summary>
        /// Méthode qui va insérer une réponse ouverte dans la base de données selon un object
ReponseOuverte
        /// </summary>
        /// <param name="r">object ReponseOuverte</param>
        public void insertReponseOuverteObject(ReponseOuverte r)
        {
            proxy = new QuestionnaireContractClient();
            try
            {
                proxy.insertReponseOuverteObject(r);
            }
            catch (FaultException<QuestionnaireServiceException> error)
            {
                ErrorMessageBox.Show(error.Detail.text, error.Detail.details, error.Detail.title);
            }
            catch (EndpointNotFoundException ex)
            {
                ErrorMessageBox.Show("L'application n'a pu trouver le webservice. Vérifiez la
configuration du endpoint dans le fichier App.config.", ex.Message, "Endpoint non trouvé");
            }
            catch (Exception ex)
            {
                ErrorMessageBox.Show("exception rencontrée", ex.Message, "erreur dans le web
service");
            }
        }


        /// <summary>
        /// Méthode qui va insérer une réponse à une fiche de résultat
        /// Cette action intervien lorsqu'une personne choisi une réponse à un questionnaire
        /// </summary>
        /// <param name="idFicheResultats">indentifiant d'une fiche de résultats</param>
        /// <param name="idReponse">identifiant d'une réponse</param>
        public void insertReponseQuestionnaire(int idFicheResultats, int idReponse)
        {
            proxy = new QuestionnaireContractClient();
            try
            {
                proxy.insertReponseQuestionnaire(idFicheResultats, idReponse) ;
            }
            catch (FaultException<QuestionnaireServiceException> error)
            {
                ErrorMessageBox.Show(error.Detail.text, error.Detail.details, error.Detail.title);
```

```
            }
            catch (EndpointNotFoundException ex)
            {
                ErrorMessageBox.Show("L'application n'a pu trouver le webservice. Vérifiez la
configuration du endpoint dans le fichier App.config.", ex.Message, "Endpoint non trouvé");
            }
            catch (Exception ex)
            {
                ErrorMessageBox.Show("exception rencontrée", ex.Message, "erreur dans le web
service");
            }
        }
    }
}
```

Cette classe contient l'équivalent des méthode que l'on trouve dans le ContractType du web service. Néanmoins dans un but d'utilisation pour le client.

Si on regarde de plus près ces méthodes, on distingue la composition suivante:

- Création d'un nouveau proxy pour interroger le web service
- Interrogation du web service
- Gestion des erreurs

Il est à noter que le proxy nommé « QuestionnaireContractClient » a été généré avec l'outil svcutil.

Pour ce qui est de la gestion des exception, trois types sont récupérés:

- FaultException<QuestionnaireServiceException>

  Ce sont les exceptions générées par le web service.
- EndpointNotFoundException

  Ce sont des exceptions lorsque le client ne trouve pas le web service.
- Exception

  Récupération de toutes les autres exceptions.

Vous remarquez que lorsque le client rencontre une exception, il affiche un formulaire « ErrorMessageBox ». Ce formulaire a été conçu pour l'affichage des erreurs et fait partie du projet « QuestionnaireErrorMessageBox ».

5.6. Développement Concret basé sur le projet Questionnaire pour analyses

Le développement concret s'est basé sur une analyse, du magazine psychologies du mois de novembre 2006.

Le titre de cette analyse est :

Êtes-vous acteur ou spectateur de votre vie ?

Et son but est d'établir un profile psychologique pour une personne qui a été interrogée sur une série de questions.

Voici les éléments que j'ai créés pour mettre en place cette analyse :

- scripte pour l'insertion des valeurs test dans la base de données.
- un module métier pour analyser les réponses
- un client win32 pour réaliser le questionnaire

Voici un schéma de l'application :

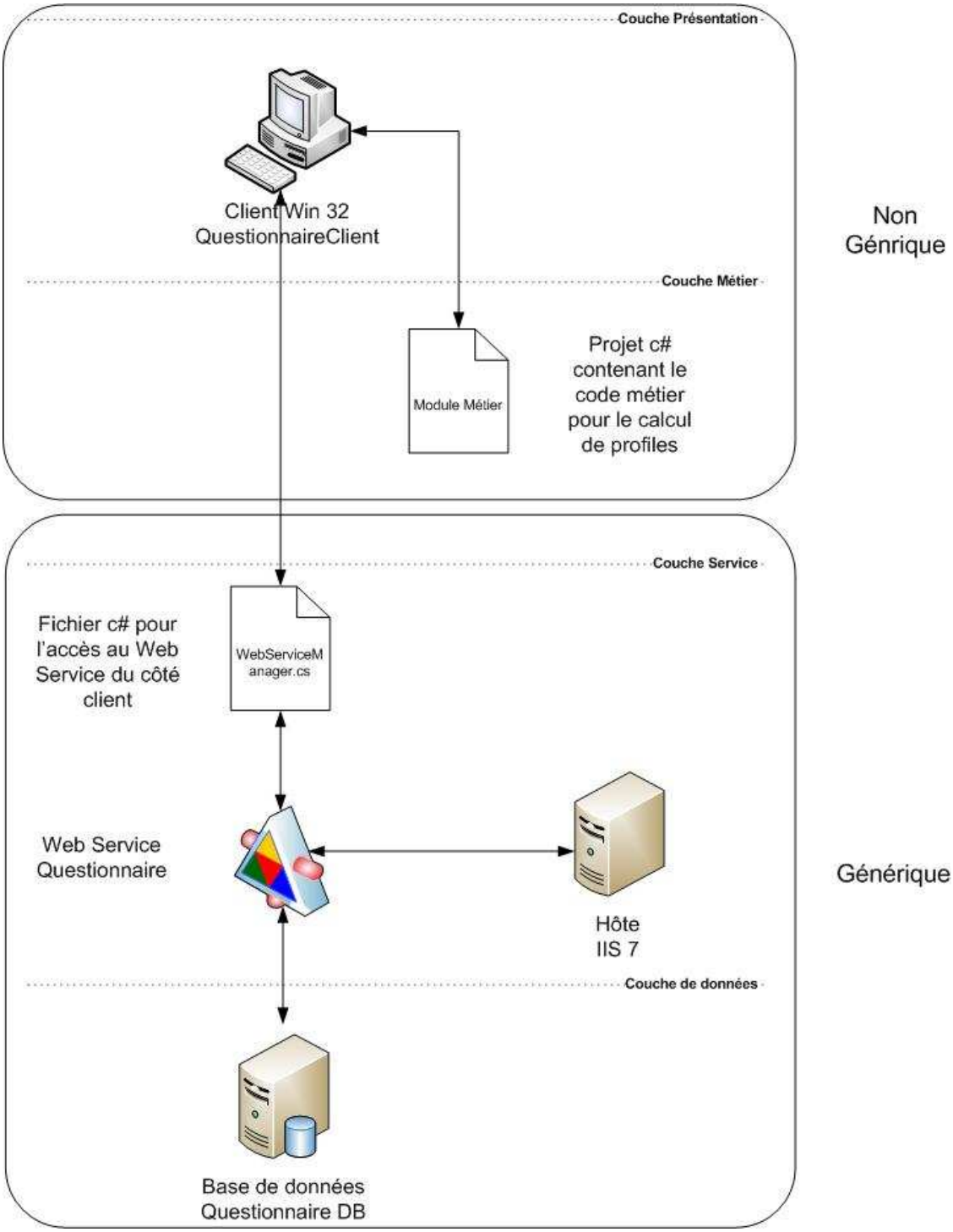


Schéma de l'exemple concret basé sur le projet Questionnaire pour analyses

Nous remarquons bien la distinction dans ce schéma de la partie générique et non-générique.

Concrètement, le Client Win32, le Module Métier et le fichier WebServiceManager, se trouvent dans trois projets c# distincts.

Ces projets se trouvent dans la même solution et à leur compilation génèrent le client « final » pour la réalisation du questionnaire.

Vous trouverez les scriptes pour l'ajout de données dans le cd de ressources WCF.

Répertoire: ProjetQuestionnaire\ExempleConcret\ScriptsSQL

Module métier

Le module métier est composé de deux classes :

- FicheResultats
- Profile

Voici la composition de la clase Profile :

```
using System;
using System.Collections.Generic;
using System.Text;
using QuestionnaireClientTools;
using System.Data;

namespace ModuleMetier
{
   public class FicheResultats
   {
      /// <summary>
      /// variables
      /// </summary>
      private int idPeronne ;
      private int idQuestionnaire ;
      private int idFicheResultats ;

      /// <summary>
      /// constructeurs
      /// </summary>
      /// <param name="idPersonne">identifiant d'une personne</param>
      public FicheResultats(int idPersonne)
      {
         this.idPeronne = idPersonne ;
      }

      /// <summary>
      /// Getters Setters
      /// </summary>
      public int IDPersonne
      {
         get { return idPeronne;}
         set { idPeronne = value; }
      }

      public int IDQuestionaire
      {
         get { return idQuestionnaire; }
         set { idQuestionnaire = value; }
      }

      public int IDFicheResultats
      {
         get { return idFicheResultats; }
         set { idFicheResultats = value; }
      }
   }
}
```

Cette classe, va permettre à l'application cliente de gérer plus facilement le déroulement du questionnaire. Un objet « FicheResultats » va enregistrer localement l'identifiant du client, du questionnaire et de la fiche de résultats.

Voici le code de la classe profile :

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Xml;
using QuestionnaireClientTools;
using System.Data;

namespace ModuleMetier
{
   public class Profile
   {
      /// <summary>
      /// variables
      /// </summary>
      private string titre;
      private string description;
      private string conseil;

      private string docName = "profiles.xml";
      private string node1 = "Questionnaire";


      /// <summary>
      /// constructeur qui va créer un profile selon une fiche de résultats
      /// </summary>
      /// <param name="fr"></param>
      public Profile(FicheResultats fr)
      {
         new Profile();
         this.calculProfile(fr);
      }


      /// <summary>
      /// constructeur pour un profile vide
      /// </summary>
      public Profile()
      {
         this.titre = "";
         this.description = "";
         this.conseil = "";
      }

      /// <summary>
      /// Getters Setters
      /// </summary>
      public string Titre
      {
         get
         {
            return titre;
         }
         set
```

```
    {
        titre = value;
    }
}

public string Description
{
    get
    {
        return description;
    }
    set
    {
        description = value;
    }
}

public string Conseil
{
    get
    {
        return conseil;
    }
    set
    {
        description = value;
    }
}



/// <summary>
/// Méthode qui va calculer un profile selon une fiche de résultats
/// </summary>
/// <param name="fr">Fiche de résultats</param>
public void calculProfile(FicheResultats fr)
{
    int[] pointsProf = {0,0,0,0} ;

    WebServiceManager wsrm = new WebServiceManager();
    DataSet dsFicheResultat = wsrm.getFicheResultats(fr.IDFicheResultats);

    //calculs des points
    for (int i = 0; i < dsFicheResultat.Tables[0].Rows.Count; i++)
    {
        int idReponse = Convert.ToInt32(dsFicheResultat.Tables[0].Rows[i][1]);
        int point = wsrm.getPointReponse(idReponse);

        //incrémentation des points
        switch (point)
        {
            case 1: pointsProf[0]++;
                break;
            case 2: pointsProf[1]++;
                break;
            case 3: pointsProf[2]++;
                break;
            case 4: pointsProf[3]++;
                break;
        }
    }
```

```
            //choix du profile
            int max = pointsProf[0];
            int indprofile = 1;
            for (int i = 1; i < pointsProf.Length; i++)
            {
                if (max < pointsProf[i])
                {
                    max = pointsProf[i];
                    indprofile = i + 1;
                }
            }

            //chargement du profile
            chargeProfile(fr.IDQuestionaire, indprofile);
        }


        /// <summary>
        /// Méthode qui va lire le fichier xml des profiles et remplir l'objet en cours
        /// </summary>
        /// <param name="idProfile">identifiant du profil</param>
        public void chargeProfile(int idQuestionnaire, int idProfile)
        {
            //définition des paramètres du fichier
            XmlDocument xmldoc = new XmlDocument();
            xmldoc.Load(docName);

            XmlNodeList xmlnodeL = xmldoc.GetElementsByTagName(node1) ;
            XmlAttributeCollection xmlattr = xmlnodeL[0].Attributes;

            //parcours du fichier
            for (int i = 0; i < xmlnodeL.Count; i++)
            {
                if (Convert.ToInt32(xmlattr[0].Value) == idQuestionnaire)
                {
                    xmlnodeL = xmlnodeL[0].ChildNodes;
                    XmlNode node = xmlnodeL[0].ChildNodes[idProfile-1];

                    this.titre = node["titre"].InnerText;
                    this.description = node["description"].InnerText;
                    this.conseil = node["conseils"].InnerText;
                }
            }

        }

        /// <summary>
        /// Méthode pour l'affiage d'un profile
        /// </summary>
        /// <returns>retounrn une chaine de charactères</returns>
        public string toString()
        {
            return "Titre du profile \t" + this.titre + "\n\n" + "Description du Profile \t" + this.description +
"\n\n" + "Conseils : \n" + this.conseil;
        }
    }
}
```

Cette classe est utilisée pour la gestion des profiles.

Elle contient une méthode qui va calculer l'identifiant du profile. Une autre méthode va récupérer dans un fichier XML le contenu du profile (tire, description,conseil) selon son identifiant.

Vous trouverez le code du module métier dans le cd de ressources WCF.

Répertoire:
\ProjetQuestionnaire\ExempleConcret\QuestionnaireClientSolution

Client pour l'analyse

Le client était la phase finale pour la réalisation de mon exemple concret.

Son but, était simplement de démontrer le fonctionnement de l'application générique de base. Aucune autre fonctionnalité n'était donc requise.

Du a des contraintes de temps et d'utilisation, il a donc été défini que ce client allait être :

- une application win32.
- limité aux fonctions de base pour exécuter un exemple.

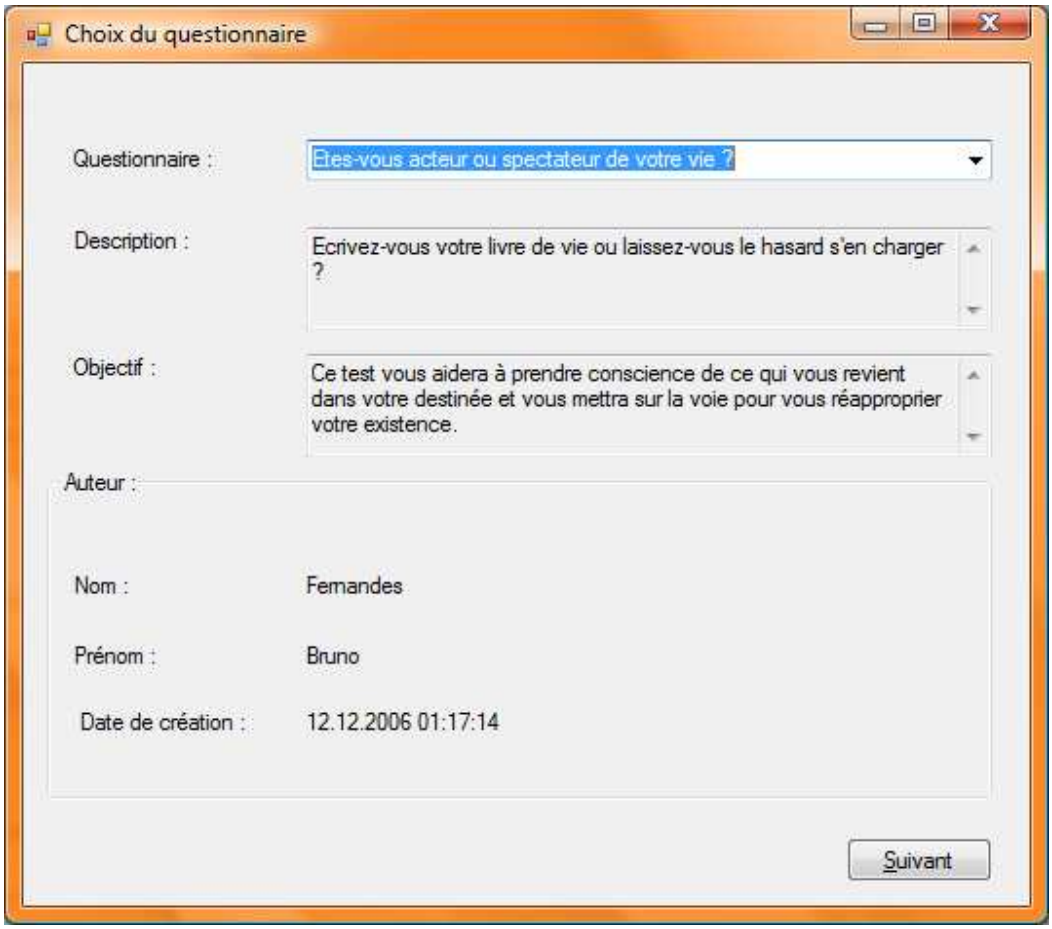
Pour avoir une version informatique de cette application consultez le cd de ressources WCF.

Répertoire:
\ProjetQuestionnaire\ExempleConcret\QuestionnaireClientSolution

Le projet du client pour l'analyse est composé des trois formulaires (classes) suivants :

- FormChoixQuestionnaire

  C’est le formulaire d’entrée de l’application. Il permet de choisir le questionnaire à exécuter.



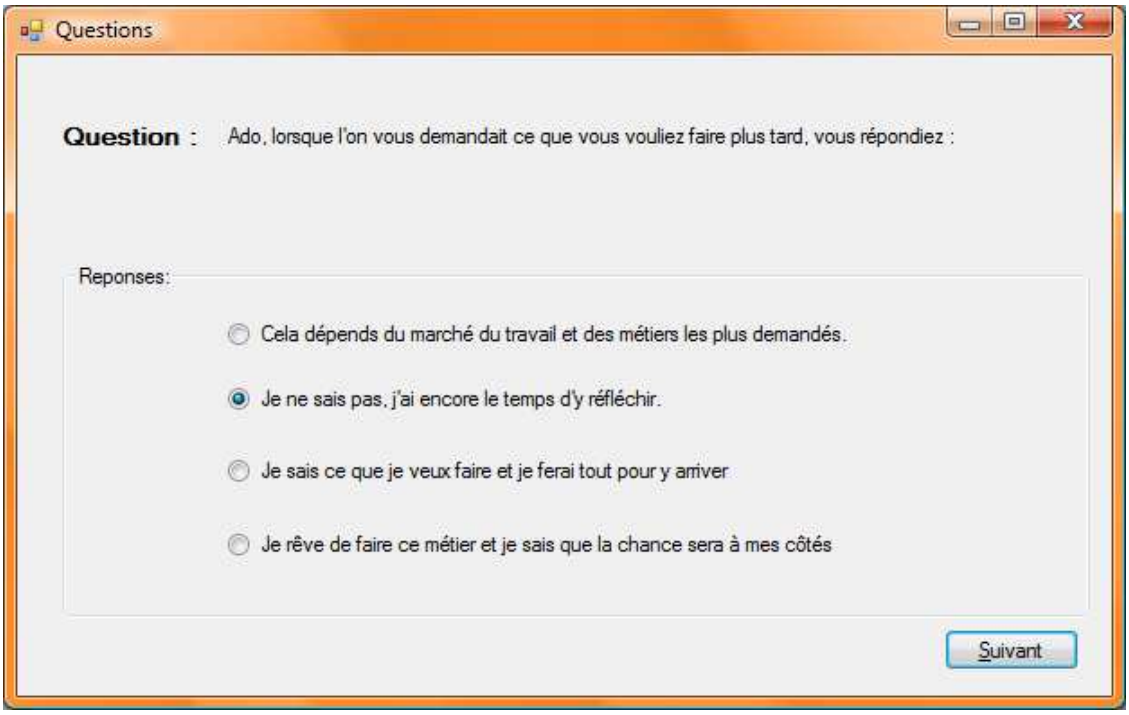
Formulaire : FormChoixQuestionnaire

En cliquant sur le bouton« Suivant » une nouvelle fiche de résultat est créée.

L’application exécutera ensuite le formulaire des questions.

- FormQuestions

  Ce formulaire va afficher les questions et réponses du questionnaire.



Formulaire : FormQuestions

En cliquant sur le bouton« Suivant » le questionnaire va enregistrer le choix et passer à la question suivante.

Arrivé à la fin des questions, le calcul du profile intervient selon le module métier. Après cette étape, le module métier va récupérer le contenu du profile (tire, description, conseil) dans un fichier XML.

L'application exécutera ensuite le formulaire qui va afficher le résultat de l'analyse.

- FormProfile

  Ce formulaire va afficher le profile obtenu pour le questionnaire qui vient de prendre fin.

Formulaire : FormProfile

En cliquant sur le bouton « quitter » le résultat au questionnaire sera enregistré dans la base de données via le web service.

6. Conclusion

6.1. Evaluation de WCF

J'ai évalué WCF selon plusieurs critères :

Installation et Mise en route à WCF

J'ai trouvé la mise en route à WCF assez simple et intuitive.

L'installation des composants nécessaires au développement a évolué de manière très positive.

Son seul défaut reste le temps requis à l'installation (près d'une heure).

Installation et Mise en route à WCF		★★★★★
Installation d'un poste pour le développement avec WCF	★★★★☆	
Mise en route à WCF*	★★★★★	

*Création d'un simple web service communicant avec basicHttpBinding, exposition d'une classe en tant que web service

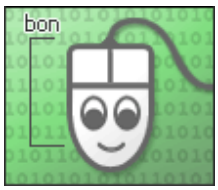
Utilisation avancée de WCF

La mise en route avancée à WCF demande beaucoup plus de temps et présente quelques difficultés. Toutefois, il faut rester conscient, qu'en maitrisant WCF, cela revient à maitriser la plus grande partie des API de communication du monde .Net.

Concernant les ressources disponibles sur internet, elles devraient rapidement être de meilleure qualité et plus nombreuses. De plus, les exemples provenant des versions « beta » devront probablement être mis à jour.

La mise en place de la sécurité est peu compliquée et une fois de plus, est gérée au niveau du fichier de configuration.

Finalement, pour ce qui est de l'hébergement d'un web service WCF, il se fait de manière très simple. Que ce soit avec une application console ou en utilisant IIS.

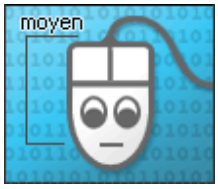
Utilisation avancée de WCF		★★★★
Mise en route avancée à WCF**	★★☆☆☆	
Possibilités de communications offertes	★★★★★	
Ressources « actuelles » à disposition (documentation et aide)	★★☆☆☆	
Hébergement d'un web service	★★★★★	
Sécurisation d'un web service	★★★★☆	

**Connaissance de l'utilisation des principaux bindings de communication. Paramétrage correcte des bindings configuration. Utilisation des outils de diagnostic. Utilisation correcte des paramètres avancés des service behaviors.

Mise en place des fonctionnalités avancées

L'implémentation des technologies avancée (reliable-session, transactions, queues) n'est pas très compliquée.

Toutefois, elle demande de réécrire en partie le code de l'application. Principalement, lors de la mise en place des queues où les méthodes qui ne sont pas one-way devront être réécrites.

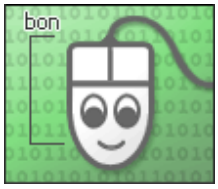
Fonctionnalités avancées		★★★
Mise en place des reliable sessions	★★★★☆	
Mise en place des transactions	★★★☆☆	
Mise en place des Queues	★★☆☆☆	

Interopérabilité

L'interopérabilité avec ASMX est excellente.

Les quelques tests avec ave le langage de programmation Java, se sont montrés concluants et ne m'ont pas posé de problèmes.

L'intégration avec Com+ est également peu compliquée. Néanmoins, le nombre de contraintes pour l'exposition des méthodes en réduit son avantage.

Interopérabilité		★★★★
Intégration avec web service ASMX	★★★★★	
Intégration avec les web service Java Sun	★★★★☆	
Intégration avec Com+	★★★☆☆	

6.2. Commentaire personnel

Jusqu'à présent, le choix du mode de communication des applications était établi selon plusieurs critères (demande du client, type d'application, protocole de communication, lieu de communication, etc.).

Penser à changer par après, le type de communication d'une application, sous-entendait automatiquement des coûts de modifications élevés. Ce choix était donc crucial !

Certaines tâches qui pouvaient sembler impossibles à effectuer ou qui demandaient un budget inestimable, deviennent maintenant un jeu d'enfant.

WCF permet cela, car il englobe la pluparts des API de communication .Net existantes, ce qui garantit une très bonne interopérabilité avec l'existant.

Après avoir travaillé trois mois sur cette nouvelle technologie, je suis toujours bluffé des possibilités qu'elle offre.

Le paramétrage de la communication dans le fichier de configuration va très loin. Il suffit de lancer l'utilitaire SvcConfigurationEditor pour s'en rendre compte.

Malheureusement, ceci apporte également un désavantage. Celui de compliquer parfois ce qui pourrait-être simple.

En effet, vu l'étendue des paramètres (bindings, bindings configuration, service behaviors, etc.), il faudrait passer de nombreux mois pour les maitriser, comprendre leur utilité et fonctionnement.

Et sans apprentissage approfondi de ses fonctionnalités, une mauvaise utilisation pourra être faite ou des problèmes pourront rapidement intervenir.

Une fois ces pré-requis mis de côté, WCF devient une arme redoutable pour la communication des applications.

D'un point de vu industriel, Windows Communication Foundation présente forcement beaucoup d'avantages. Les seules barrières concrètes représentent :

- La facilité et la rapidité d'apprentissage des développeurs.

- La mise à jour des postes de travail.

Pour conclure, Windows Communication Foundation est présenté par Microsoft, comme le fer de lance d'une nouvelle génération de communication dans les applications .Net.

N'ayant pas peur de le dire, WCF ne déçoit pas ! Et les spécifications présentées par le géant de Redmond sont belles et bien là !

6.3. Commentaire sur le tutoriel d'apprentissage

Analyser, développer, tester et rédiger. Voilà le travail qui m'a été demandé de faire pour mon tutoriel d'apprentissage.

Autant être clair, la rédaction de ce tutoriel n'a pas été facile et n'a pas été le travail le plus agréable que j'ai eu à faire.

La quantité de matière à traiter était pour un travail de 3 mois quasi-insurmontable et la réalisation des exemples ne s'est pas toujours réalisée sans encombre.

De plus, après la création de chaque nouvel exemple, il fallait l'intégrer au tutoriel. Un grand travail de relecture était alors également demandé.

Travailler avec une technologie en développement a aussi posé beaucoup de contraintes, bien que ce soit toujours quelque chose de fort intéressant.

Il a donc fallut faire des concessions au niveau de la matière traitée et fixer des priorités.

Certains chapitres, tels que l'utilisation des communications pair à paire (peer to peer) ou l'intégration de WCF à des composants com+ utilisant msmq non donc pas été traités. Les chapitres non traités seraient donc des améliorations envisageables à ce tutoriel.

6.4. Commentaire sur le développement concret WCF

J'ai trouvé le sujet de mon développement concret intéressant. A première vue, il m'a paru peu évident de réaliser un web service générique qui pourrait satisfaire à plusieurs analyses.

Mais après un grand travail de recherche, il est devenu beaucoup plus clair, comment allait être la structure de mon application.

Pour ce qui est du développement, il n'a pas été très compliqué. Néanmoins, il a requis beaucoup de temps, ceci dû aux nombreuses méthodes exposées.

La prochaine amélioration, serait la création d'un générateur de web service. Toutefois il reste très difficile de créer un service réutilisable sans y mélanger un peu de logique métier.

Remerciements

Je tiens à remercier tout d'abord Jean-Pierre Rey qui a été mon responsable de Travail de diplôme. Il a toujours été disponible et m'a apporté de la « précieuse » documentation pour la réalisation de mon tutoriel.

Remerciement également à :

- Bruno Montani :
    - responsable de la filière informatique
- Claude-Alain Emery :
    - responsable des informations relatives aux travaux de diplôme
- David Russo :
    - responsable de la configuration des machines à disposition pour la réalisation des travaux de diplôme.

7. Sources

7.1. Sites Webs

Site contenant plusieurs tests psychologiques.	http://www.psychologies.com
Construction d'un Qcm	http://tecfa.unige.ch/tecfa/teaching/staf16/qcm.html
site officiel de WCF	http://wcf.netfx3.com
Introduction à WCF	http://www.falafel.com/Community/blogs/wcf/default.aspx
Base de programmation pour WCF	http://www.microsoft.com http://msdn2.microsoft.com/en-us/library/aa388579.aspx
Présentation WINFX & WCF	http://www.dotnet-news.com/lien.aspx?ID=19254
Activer la publication des métadonnées	http://www.codeproject.com/winfx/WCFServiceClient.asp http://msdn2.microsoft.com/en-us/library/ms734765.aspx
WCF Transactions (WS-AT)	http://windowssdk.msdn.microsoft.com/en-us/library/aa347993.aspx http://windowssdk.msdn.microsoft.com/en-us/library/ms733904.aspx http://windowssdk.msdn.microsoft.com/en-us/library/ms730232.aspx

WCF Virtual Labs	http://msdn.microsoft.com/virtuallabs/wcf/
Sécuriser un web service WCF	http://www.theserverside.net/tt/articles/showarticle.tss?id=SecuringWCFService
WCF FaultException	http://forums.microsoft.com/MSDN/ShowPost.aspx?PostID=728236&SiteID=1
Choisir un transport dans WCF	http://www.theserverside.net/tt/articles/showarticle.tss?id=SecuringWCFService
Migration d'une application .Net Remoting à WCF	http://www.techheadbrothers.com/Articles.aspx?id=3b07ebc5-545e-4e33-aaa4-80f94ce1934b
WCF et Reliable Messaging	http://wcf.netfx3.com/content/IntroductiontoReliableMessagingwiththeWindowsCommunicationFoundation.aspx
Communication Two-Way avec MSMQ	http://msdn2.microsoft.com/en-us/library/ms752264.aspx
Utilisation des procédures stockées avec .Net	http://www.aspfr.com/codes/UTILISATION-PROCEDURE-STOCKE-AVEC-SQLSERVER_8953.aspx http://www.aspfr.com/code.aspx?ID=19515
Users controls en C#	http://morpheus.developpez.com/usercontrols/
Utilisation des fichiers xml en c#	http://csharpro.developpez.com/page4.php

7.2. Blogs

Bindings dans WCF	http://blogs.msdn.com/drnick/archive/2006/07/19/670789.aspx
java se met à WCF (Projet Tango)	http://weblogs.java.net/

7.3. Webcast

Sécurisation des services Web avec WSE 3.0 et WCF	Webcast WSE 3.0 WCF 1 Webcast WSE 3.0 WCF 2 Webcast WSE 3.0 WCF 3
Webcast WCF	Webcast WCF 1 Webcast WCF 2 Webcast WCF 3 Webcast WCF 4 Webcast WCF 5

7.4. Livres

Microsoft Windows Communcaiton Foundation Hands-on ! Beta Edition	Sams www.samspublishing.com
WinFx Beta	Wrox http://www.wrox.com

7.5. Magazines

Psychologies	Novembre 2006

8. Annexes

Vous trouverez en annexe à ce rapport les documents suivants :

- Cahier des charges du Tutoriel WCF
- Cahier des charges du développement concret WCF
- Tutoriel d'apprentissage WCF